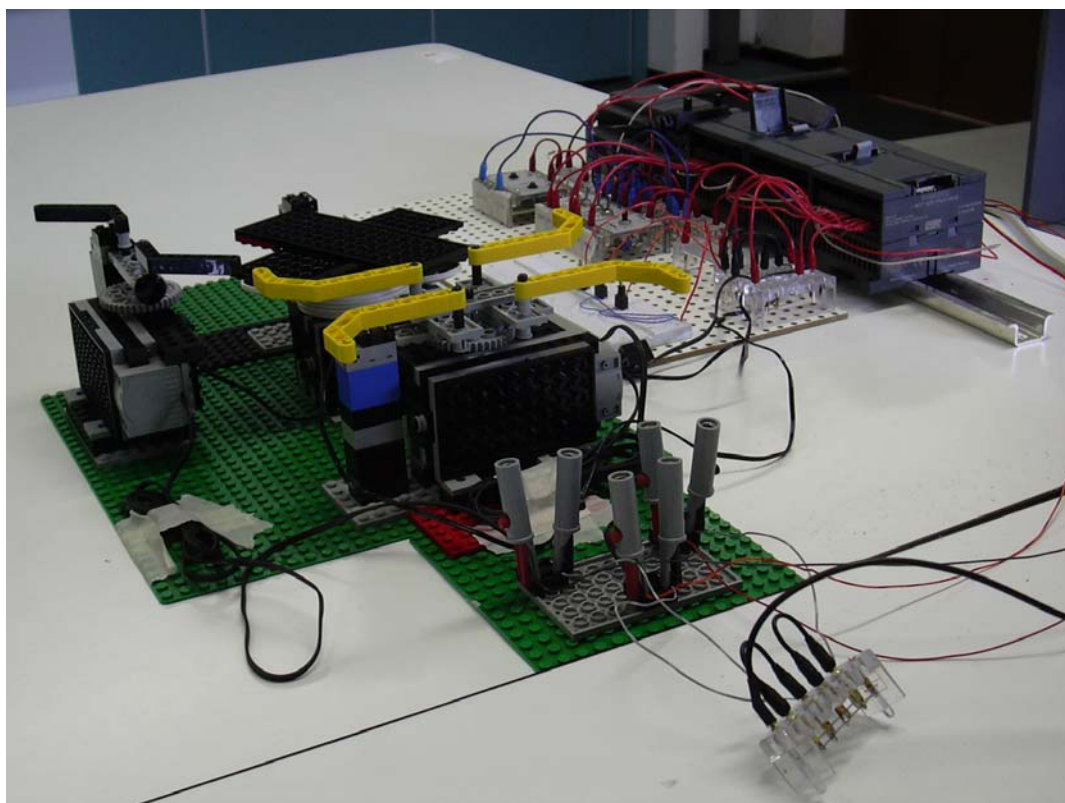
Politecnico di Milano – Sede di Cremona
Anno Accademico 2002/2003

LAUREA DI PRIMO LIVELLO IN INGEGNERIA INFORMATICA

- PROVA FINALE -

ESPERIMENTI DI CONTROLLO LOGICO CON *PLC* ED E COMPONENTI *LEGO MINDSTORMS*

Relatore:

Prof. Alberto Leva

Autori:

Frigoli LucaGregorio matr. 646342
Lapris Stefano matr. 646256

“ [La scienza] ci insegna a non trascurare niente, a non disdegnare gli inizi modesti…in quanto nel piccolo sono sempre presenti i principi del grande, come nel grande è contenuto il piccolo”

.

- *Michael Faraday –*

a NOI

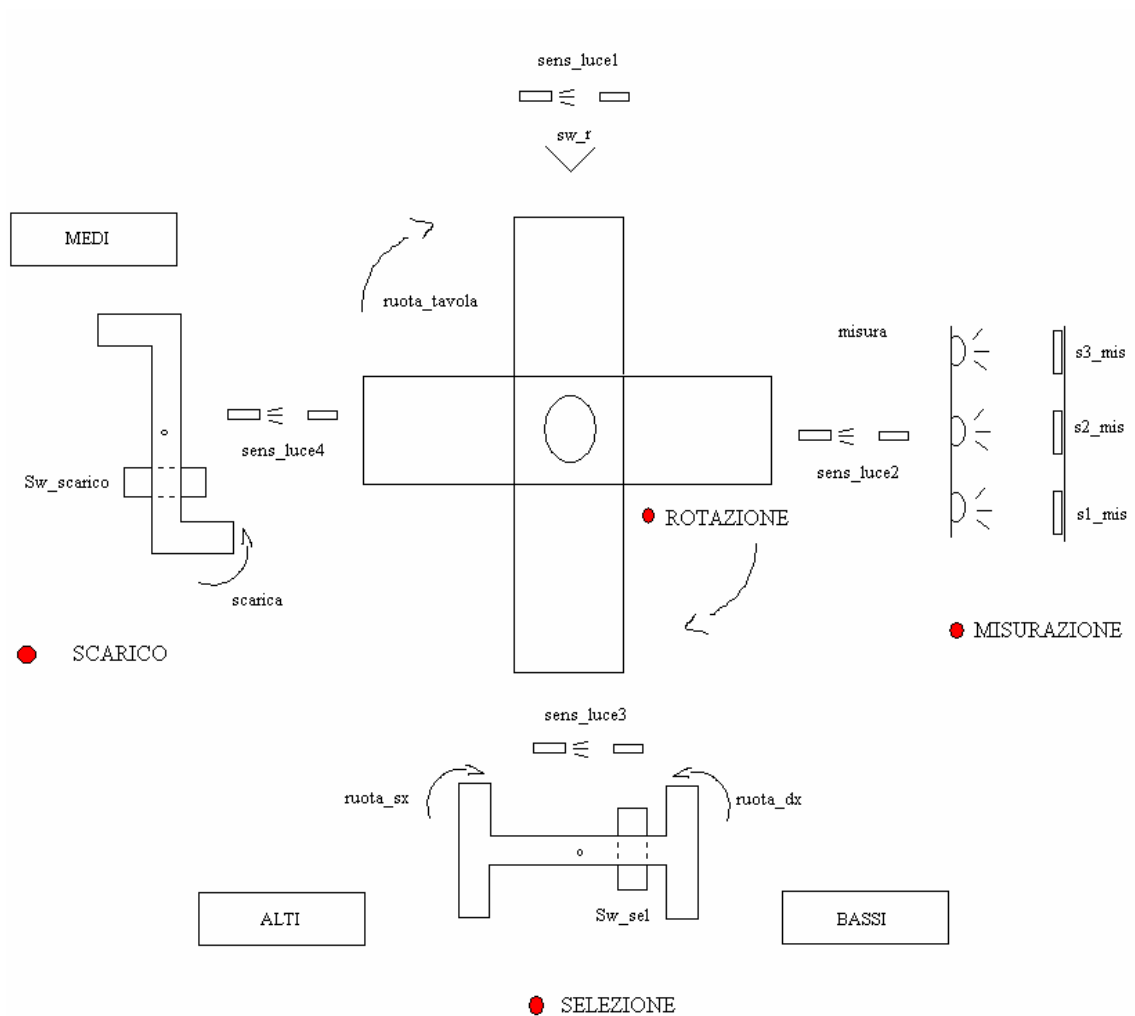
Ste e Greg

OBIETTIVI

L'obiettivo principale è quello di creare un esperimento didattico, facilmente riproducibile da uno studente universitario, consistente nella creazione del prototipo di una macchina industriale da controllare mediante un controllore a logica programmabile (PLC).

La macchina industriale scelta è costituita dai seguenti componenti:

1. una stazione centrale di rotazione che permette di spostare i pezzi presenti verso le altre stazioni
2. una stazione di misura con la quale si determina il tipo di pezzo presente in base alla sua altezza
3. una stazione di selezione con cui si separano i pezzi bassi da quelli alti
4. una stazione di scarico con la quale si scaricano dalla tavola rotante i pezzi di media altezza

Le stazioni 2,3 e 4 sono disposte tra loro a distanza di 90° rispetto al centro della tavola rotante, elemento principale della stazione di rotazione, costituita da quattro braccia.
E' facile intuire che è quindi presente una postazione "vuota" la quale rappresenta la postazione di carico che nell'esperimento viene considerato effettuato in modo corretto ed idoneo da parte di un operatore qualsiasi sia esso umano o robotizzato.

L'evoluzione dell'esperimento si articola sostanzialmente in tre fasi:

a) costruzione fisica del prototipo
b) simulazione del funzionamento mediante software dedicato
c) funzionamento reale

Queste costituiscono gli argomenti principali dei prossimi capitoli nei quali vengono descritti nel dettaglio.

Capitolo 1

Progettazione del Prototipo LEGO

La progettazione di un prototipo lego viene ottimamente descritta dal modello virtuale il quale riproduce in modo chiaro e visivo l'evoluzione della costruzione dal primo all'ultimo pezzo, evidenziandone i passi salienti ed avendo ogni volta un riscontro percepibile.

Il modello virtuale consiste in un insieme di istruzioni grafiche alle quali viene aggiunta una tabella riassuntiva di tutti i pezzi utilizzati e delle proprie caratteristiche, raggruppati in base ai passi da eseguire.

Tutto ciò è rappresentato nel fascicolo allegato:

" ***PROTOTIPO LEGO - Manuale di istruzioni*** "

nel quale sono rappresentate le istruzioni grafiche e le tabelle dei pezzi utilizzati per ogni stazione di lavorazione costituenti l'intera macchina, nonché la disposizione finale di quest'ultime.

NOTA: Le diverse istruzioni grafiche vengono create mediante un programma CAD, nello specifico MLCAD, scaricabile gratuitamente via Intenet dal sito :
www.Im-software.com/mlcad

Capitolo 2

Simulazione del funzionamento mediante ambiente ISaGRAF

2.1 INTRODUZIONE AL PROGRAMMA ISaGRAF

Ora che il prototipo lego è costruito è necessario simularne il funzionamento desiderato.
Per far questo ci si serve dell'ambiente ISaGRAF.
Si deve realizzare un programma in grado di controllare e simulare il completo funzionamento del sistema avvalendosi del linguaggio Ladder Diagram, per quanto concerne il controllore, e del linguaggio Sequential Functional Chart, per quanto riguarda il simulatore.

La scelta del linguaggio per implementare il controllore non è casuale; il linguaggio Ladder Diagram, o più semplicemente LD, è infatti molto simile al linguaggio proprietario Siemens, il KOP, che permette di programmare il controllore a logica programmabile.
Il programma deve essere sviluppato con l'intento di garantire e preservare al massimo la modularità e di conseguenza la scalabilità del modello reale.

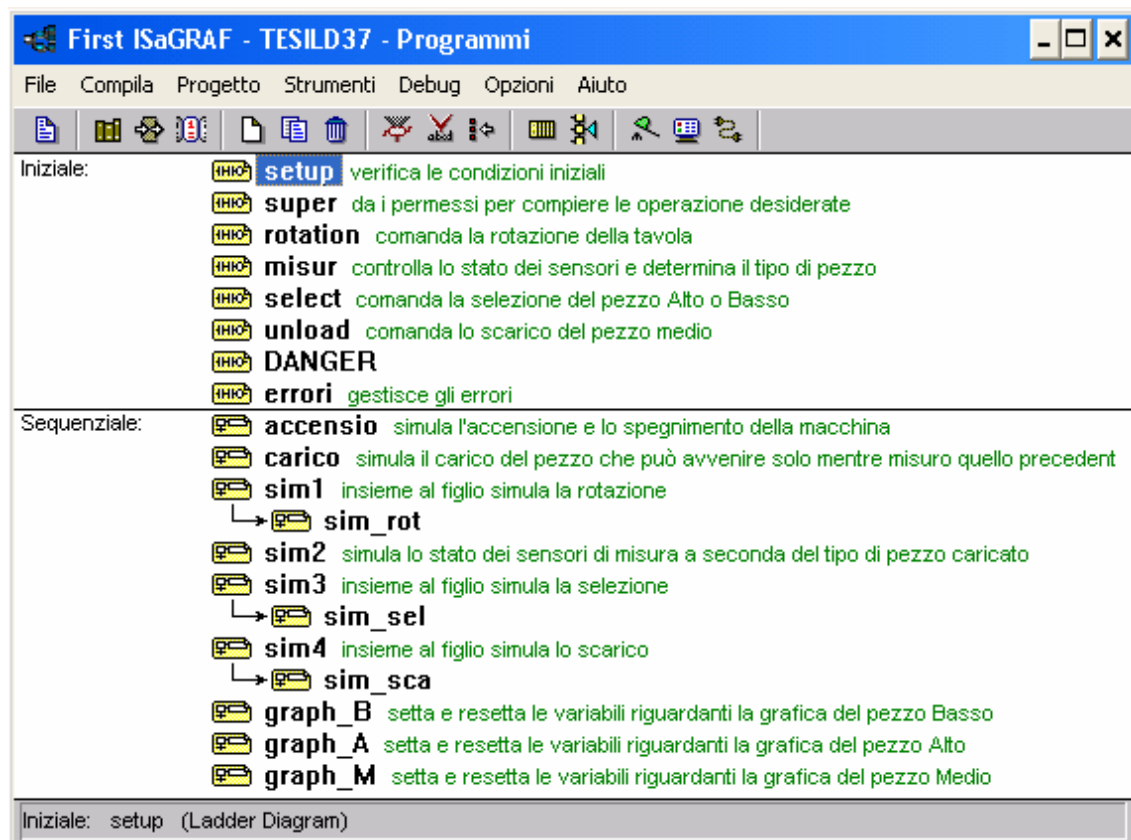



fig. 1

LISTA DELLE VARIABILI DEI PROGRAMMI DI CONTROLLO

→ SISTEMA

- **start** : variabile interna che indica l'accensione/spegnimento del sistema
- **acceso** : variabile d'uscita che accende il led corrispondente sul pannello di controllo
- **setup_ok** : variabile interna che indica la riuscita del setup
- **not_iniziali** : variabile interna che indica la non riuscita del setup
- **err_set** : variabile d'uscita che accende il led corrispondente sul pannello di controllo
- **emergency** : variabile d'ingresso che simula uno stato di emergenza e quindi il blocco istantaneo del sistema
- **perm_rot, perm_rot1, perm_mis, perm_select, perm_scarico** : variabili interne che consentono lo svolgimento delle operazioni di rotazione, misurazione, selezione e scarico
- **Com_medio, Com_ alto, Com_basso :** variabili che vengono usate per tenere memoria dei pezzi caricati nel sistema.
- **errore** : variabile interna attiva al verificasi di un malfunzionamento
- **err_rot, err_mis, err_sel, err_sca** : variabile d'uscita che accendono il led corrispondente sul pannello di controllo per indicare a quale postazione è dovuto l'errore
- **attesa** : variabile timer che indica il tempo d'attesa tra la fine di una lavorazione e l'inizio dell'operazione successiva
- **t_mis** : variabile timer che indica il tempo occorrente per la misurazione
- **t_err** : variabile timer che indica il timeout

→ STAZIONE DI CARICO

- **sens_luce1** : sensore che rileva la presenza del pezzo

→ STAZIONE DI MISURAZIONE DELL'ALTEZZA

- **s1_mis, s2_mis, s3_mis** : sensori luce che misurano l'altezza del pezzo
- **sens_luce2** : sensore che rileva la presenza del pezzo
- **MISURAZIONE**: variabile d'uscita che accende il led corrispondente sul pannello di controllo
- **fine_mis** : variabile interna utilizzata per comunicare al supervisore padre il termine della misurazione
- **Alto, Medio, Basso** : variabile interna che indica l'altezza del pezzo misurato

→ STAZIONE DI SELEZIONE DEL PEZZO

- **sens_luce3** : sensore che rileva la presenza del pezzo in posizione 3
- **SELEZIONE**: variabile d'uscita che accende il led corrispondente sul pannello di controllo
- **fine_sel** : variabile interna utilizzata per comunicare al supervisore padre il termine della selezione
- **sel_BASSO** : variabile interna che da il comando di rotazione verso destra al selettore per selezionare il pezzo Basso
- **sel_ALTO** : variabile interna che da il comando di rotazione verso sinistra al selettore per selezionare il pezzo Alto
- **sw_select** : sensore che indica la posizione del selezionatore

→ STAZIONE DI SCARICO

- **sens_1uce4**: sensore che rileva la presenza del pezzo in posizione 4
- **SCARICO** : variabile d'uscita che accende il led corrispondente sul pannello di controllo
- **fine_scarico** : variabile interna utilizzata per comunicare al supervisore padre il termine dello scarico
- **scarica** : variabile interna che da il comando di rotazione allo scaricatore
- **sw_scarico** : sensore che indica la posizione dello scaricatore

→ STAZIONE DI ROTAZIONE DELLA TAVOLA

- **Rotazione** : variabile d'uscita che accende il led corrispondente sul pannello di controllo
- **sw_r**: switch posto nella postazione di carico che serve a rilevare la rotazione della tavola
- **fine_rotation**: variabile interna utilizzata per comunicare con il supervisore padre
- **ruota_tavola**: variabile interna che da il comando di rotazione alla tavola
-

2.2 ANALISI DEL CONTROLLORE

Dalla fig.1 si può dedurre l'approccio seguito :

- ***setup*** : il programma che all'accensione controlla le condizioni dei sensori, dei comandi e degli switch
- ***super*** : il controllore vero e proprio, gestisce l'insieme delle possibili operazioni fornendo o negando i permessi
- ***rotation*** : il programma che controlla la rotazione della tavola
- ***misur*** : il programma che controlla la misurazione dei pezzi caricati
- ***select*** : il programma che controlla la selezione dei pezzi bassi o alti
- ***unload*** : il programma che controlla lo scarico dei pezzi di media altezza
- ***DANGER*** : il programma che controlla lo stato di emergenza
- ***errori*** : il programma che gestisce gli eventuali errori di ogni stazione di lavoro

➢ 2.2.1 Setup

Con il programma ***setup*** si devono verificare le condizioni iniziali, ovvero tutti i sensori luce ed i comandi disattivati e gli switch attivi, le quali vanno chiaramente controllate solo nel momento in cui la macchina viene accesa, per questo motivo nel primo ramo (fig. 1 setup) si inserisce un salto al ramo etichettato con INIZIO, (fig. 4 setup) effettuato quando la variabile *setup_ok* viene settata ovvero se non ci sono stati problemi iniziali.

setup_ok
>>INIZIO

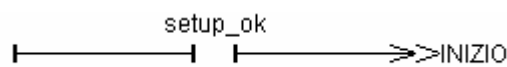



fig. 1 *setup*

La scelta per controllare le condizioni iniziali può essere duplice, dalla fig. 2 setup si deduce una logica di tipo negativo ovvero si verifica se almeno una delle condizioni iniziali è FALSA ed in tal caso si attiva la variabile *not_iniziali* che insieme alla condizione di macchina accesa attiva a sua volta *err_setup* (fig. 3 setup)

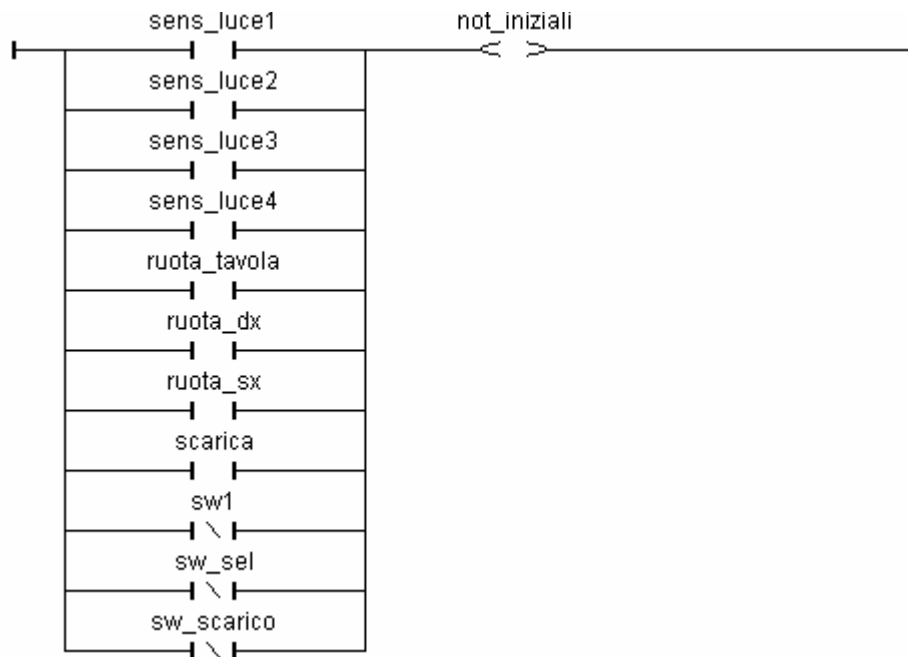



fig. 2 *setup*

NOTA:*si può usare anche una logica positiva: verificando che tutte le condizioni iniziali sono VERE si attiva una variabile 'iniziali'.*
Entrambe le scelte sono valide come dimostrato dal teorema logico di De Morgan:

$$\sim(A \wedge B) \equiv \sim A \vee \sim B$$

fig. 3 *setup*

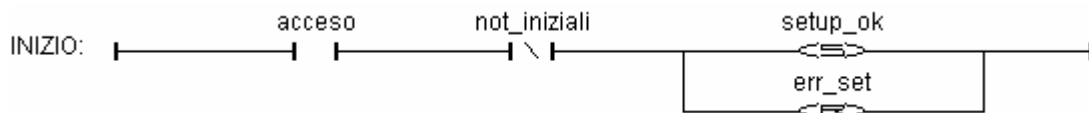



fig. 4 *setup*

L'ultimo ramo del programma (fig. 5 setup) serve a resettare la variabile *setup_ok* quando la macchina viene spenta, così che è possibile rieffettuare il controllo delle condizioni iniziali ogni qual volta la macchina venga accesa.

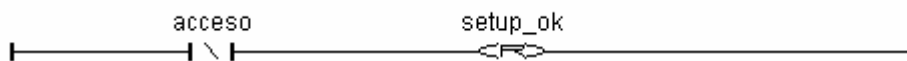



fig. 5 *setup*

➢ 2.2.2 Super

Nella figura sottostante (fig. 1 super) è rappresentata la prima istruzione del programma ***super*** con la quale si accende la macchina (bobina *acceso*) in caso di non *emergency*, non *errore* e naturalmente di ingresso *start* attivo.

In seguito la variabile *acceso* andrà posta in questo programma come prima condizione di ogni altro ramo e comparirà anche nel primo rung di tutti gli altri programmi di controllo. E' dunque chiara la sua importanza sia nel programma ***super***, essendo il primo contatto di ogni ramo, sia negli altri programmi (***rotation, misur, select, unload, DANGER, errori***), nei quali si trova in una posizione particolare che le permette di evitare il funzionamento del sistema in caso di emergenza o di spegnimento.

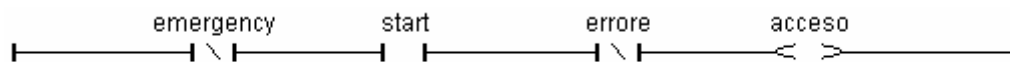



fig.1 ***super***

Attraverso il contatto negato che rappresenta la variabile *errore* si ottiene il collegamento tra il ***super*** e il programma di gestione degli errori. Qualora si verificasse un'anomalia sia nel funzionamento dell'intero sistema sia in una singola postazione di lavoro la variabile interna *errore* viene settata e di conseguenza *acceso* non può essere vera.

Oltre a questa istruzione il programma ***super*** contiene la definizione delle condizioni che consentono agli altri programmi di controllo di ottenere i permessi necessari per compiere le loro operazioni.

Vediamo ad esempio il permesso di rotazione :

Queste tre condizioni servono a sincronizzare l'operazione di rotazione con le altre in modo che la tavola non cominci a ruotare quando le altre postazioni non hanno ancora terminato.

È presente un pezzo nella posizione di carico

fig. 2 ***super***

Se è vera la variabile *acceso* significa che è stato premuto il pulsante start e non c'è emergenza

Se non ci sono più pezzi da caricare si attende che la seconda postazione abbia terminato l'operazione di misura

Un diverso permesso di rotazione è previsto nel caso in cui l'ultimo pezzo rimasto nell'impianto sia un pezzo di media altezza.

acceso Com_medio sens_luce1 sens_luce2 sens_luce3 Com_alto Com_basso sens_luce4 perm_rot1

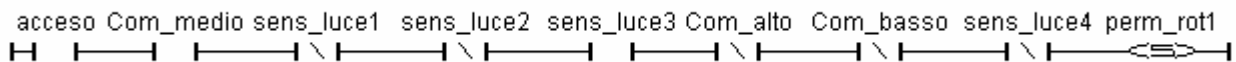



fig. 3 ***super***

Quando questo si trova in corrispondenza della terza postazione, quella di selezione, il programma da il permesso di rotazione denominato *perm_rot1*, che diversamente da *perm_rot* permette l'avvio della rotazione senza dover attendere la fine della fase corrispondente alla posizione che il pezzo occupa.

I permessi di misurazione, selezione e scarico sono praticamente identici fra loro :

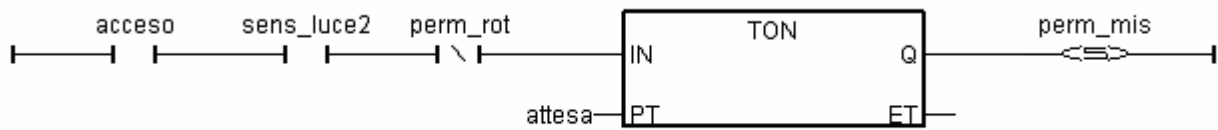



fig. 4 *super*

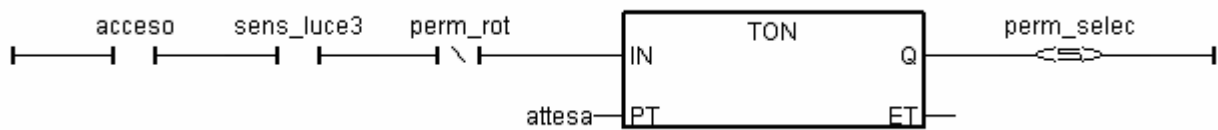



fig. 5 *super*

fig. 6 *super*

Ogni permesso viene dato quando:

- è vera la variabile *acceso* ;
- il sensore corrispondente alla postazione è attivo (per la fig. 4 la postazione 2 della misurazione, per la fig. 5 la postazione 3 della selezione ed infine per la fig. 6 la postazione 4 dello scarico), ovvero c'è un pezzo nella postazione 2 della misurazione (fig. 4) e/o un pezzo nella postazione 3 della selezione (fig.5) e/o nella postazione 4 dello scarico (fig. 6) ;
- è trascorso 1 secondo da quando la tavola ha terminato la rotazione .

Per misurare il trascorrere del tempo si utilizza il blocco TON il cui funzionamento è esemplificato nell'*Appendice A*.

➢ 2.2.3 Rotation

Con il sistema in una condizione di normale funzionamento e quindi con la variabile *acceso* vera, un secondo dopo aver ricevuto da ***super*** il permesso di ruotare (*perm_rot* o *perm_rot1*) viene acceso il led corrispondente alla rotazione e vengono resettati gli altri permessi in modo che nessuna macchina compia delle operazioni mentre la tavola sta ruotando.

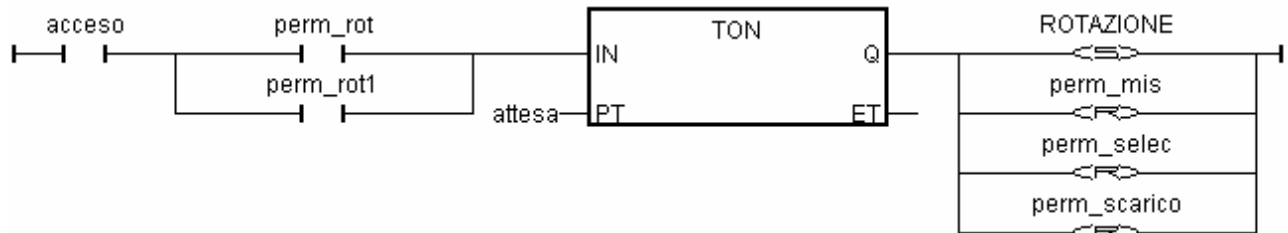



fig. 1 *rotation*

L'uso del blocco TON serve anche in questo caso per far trascorrere 1 secondo (PT è posto uguale ad *attesa*) tra la ricezione del permesso e l'accensione del led *ROTAZIONE*.
Come detto in precedenza lo stato del comando di rotazione della tavola è diretta conseguenza della condizione della la variabile *acceso* la quale è in grado di interrompere il programma (ovvero la rotazione della tavola) in caso di emergenza o spegnimento del sistema.

Se il sistema è acceso e quindi non è in stato di emergenza si riesce a settare la variabile *ROTAZIONE* e tramite questa si da il comando *ruota_tavola.*

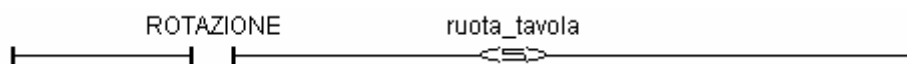



fig. 2 *rotation*

Lo switch *sw_r* collocato in corrispondenza della postazione di carico funziona come un contatto normalmente aperto: conduce corrente solo quando è chiuso (ovvero quando è premuto).
Nel momento in cui la tavola inizia a ruotare non si appoggia più allo switch che di conseguenza si apre e quindi non conduce. Torna a condurre quando la tavola, effettuata una rotazione di 90°, lo richiude.

Lo switch è dunque “l’interruttore” tramite cui si determina la fine della rotazione.

Per poter riconoscere che la tavola ha compiuto la sua rotazione si deve riuscire a rilevare il fronte di salita dello switch e ciò avviene tramite i due rami seguenti :

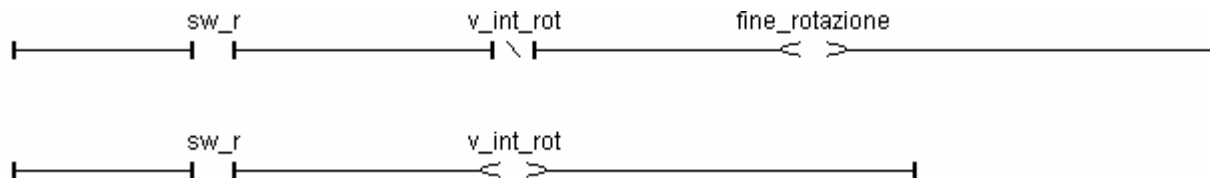
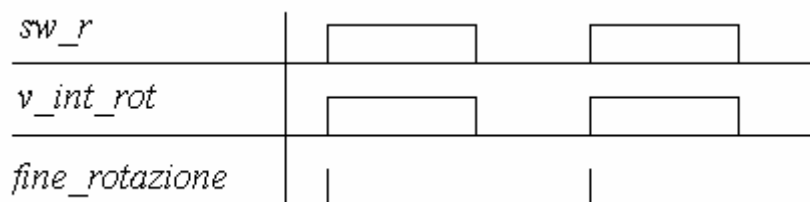



fig. 3 ***rotation***

I rami di fig. 3 rappresentano un programma che al fronte di salita dell’ingresso (*sw_r*) fa corrispondere un impulso, di durata pari a un ciclo di scansione del programma, sulla bobina *fine_rotazione* come descritto nel seguente grafico.

Quando la tavola porta di nuovo lo switch in stato di conduzione viene rilevato il fronte di salita ed attivata la variabile interna *fine_rotazione*, di conseguenza viene spento il led *ROTAZIONE* e resettati i permessi *perm_rot* e *perm_rot1*.

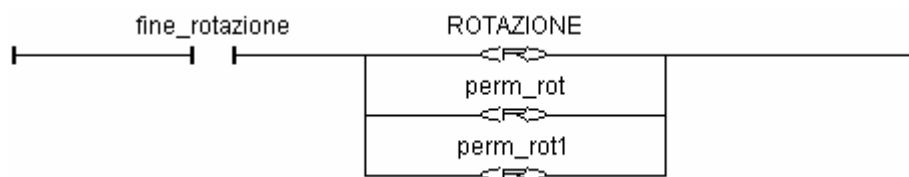



fig. 4 ***rotation***

In questo modo non c’è rischio che la tavola cominci a ruotare spostando i pezzi ancora in lavorazione.

➢ 2.2.4 Misur

Ricevuto il permesso di misurazione bisogna accendere il led *MISURAZIONE* :

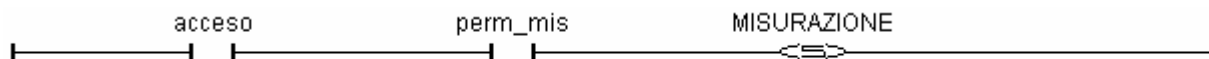
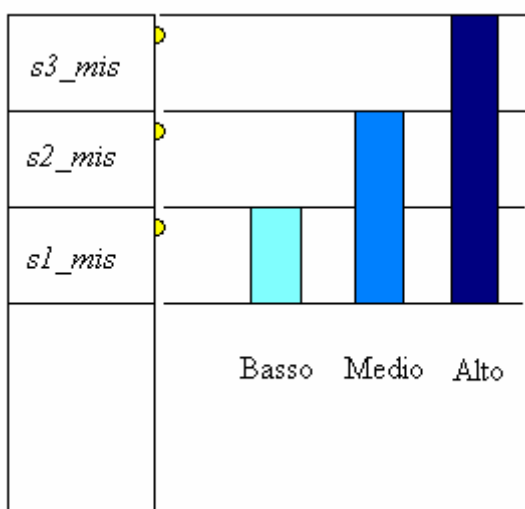



fig. 1 *misur*

Da notare che anche in ***misur*** la prima condizione del ramo è *acceso* e che quindi qualora si dovesse verificare un'emergenza il programma di misurazione resta completamente fermo .
Se *acceso* è vero si setta la variabile *MISURAZIONE* ed attraverso questa, a seconda dello stato dei sensori di misurazione, le variabili *Basso, Medio* o *Alto*. Infine grazie alla variabile che indica il tipo di pezzo si può impostare il comportamento del sistema (ciò avviene con le variabili *Com_"x"*) ed arrivare alla conclusione dell'operazione di misura.

La misurazione dei pezzi avviene tramite tre sensori di luce.

Come mostrato nella schema a lato se è attivo solo il primo sensore il pezzo è basso, mentre al contrario se lo sono tutti e tre il pezzo è alto. Se l'altezza del pezzo è media sono attivi solo i primi due sensori.

Dopo che il led è stato acceso si controlla lo stato dei sensori e, trascorso il tempo di misurazione *t_mis* posto pari a 5 secondi, viene attivata la variabile che indica l'altezza del pezzo.

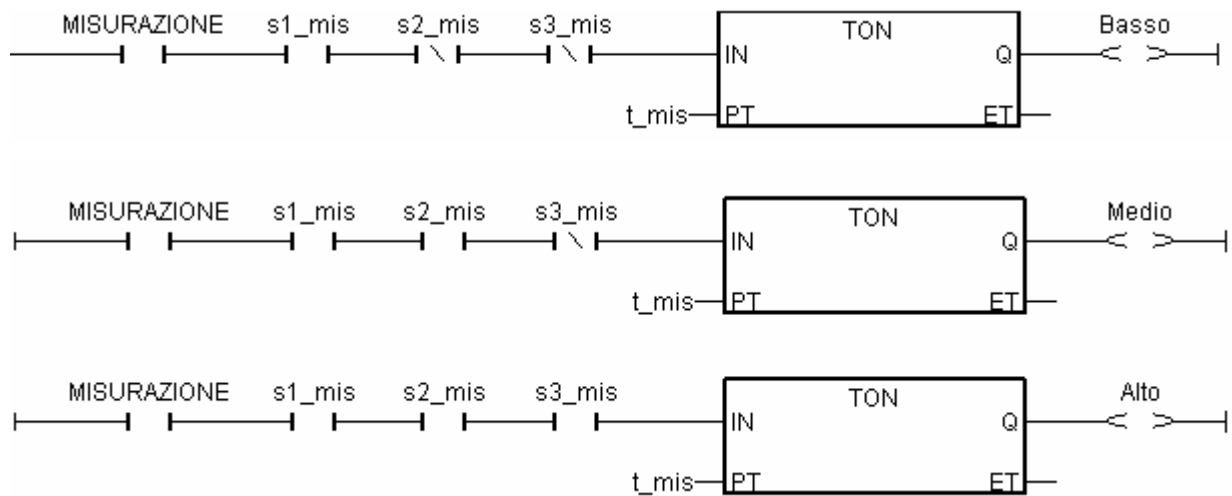



fig. 2 *misur*

Il comportamento del sistema deve variare in base al tipo di pezzo che la postazione di misura rileva; se, ad esempio, il pezzo è basso o alto attraverso le variabili *Com_basso* o *Com_alto* il sistema sa che il pezzo verrà selezionato e non arriverà alla postazione di scarico:

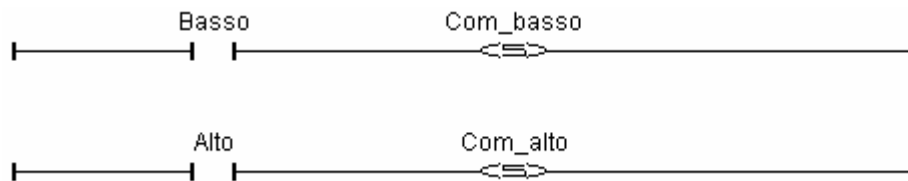



fig. 3 *misur*

Più complessa è la gestione dei pezzi di media altezza.
Per poter permettere al sistema di riconoscere i pezzi di media altezza, gli unici da ritenere adatti e da non scartare mediante l'operazione di selezione, è opportuno cercare di implementare queste poche righe di pseudocodice :

> *Inizializzazione : num_medi = 0*

> *Misurazione : num_medi = num_medi + 1*
if (num_medi >=1)
set Com_medio

> <u>Scarico</u> *: if (num_medi = 0)*

reset Com_medio

La funzione della variabile *num_medi* è estremamente importante perché grazie a questa il sistema è in grado di tenere memoria di quanti pezzi medi sono stati immessi e non incombere in errori.

La fig. 5 riportata nella pagina seguente chiarisce come deve essere implementato il codice relativo alla misurazione grazie all'uso di un blocco sommatore e di un blocco confronto.

Quando è vera la variabile *Medio* attraverso l'uso del blocco sommatore viene incrementato il numero di pezzi medi presenti nel sistema.

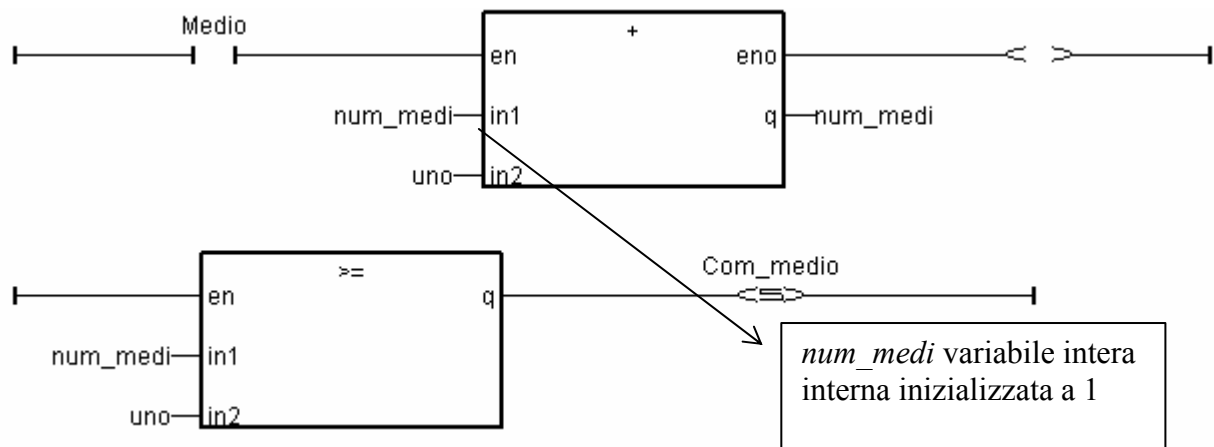



fig. 5 *misur*

Quando è presente almeno un pezzo di media altezza viene settata la variabile *Com_medio* che, come le già citate variabili *Com_basso* e *Com_alto*, identifica un determinato comportamento del sistema: il pezzo medio non verrà selezionato ma arriverà alla stazione di scarico dove verrà tolto dalla tavola rotante.

Trascorsi i 5 secondi dovuti alla variabile temporale *t_mis* ed attivata una tra le variabili *Basso, Medio* o Alto, la misurazione può considerarsi conclusa : viene attivata la variabile interna *fine_misura* e tramite questa si spegne il led corrispondente a questa stazione di lavoro.

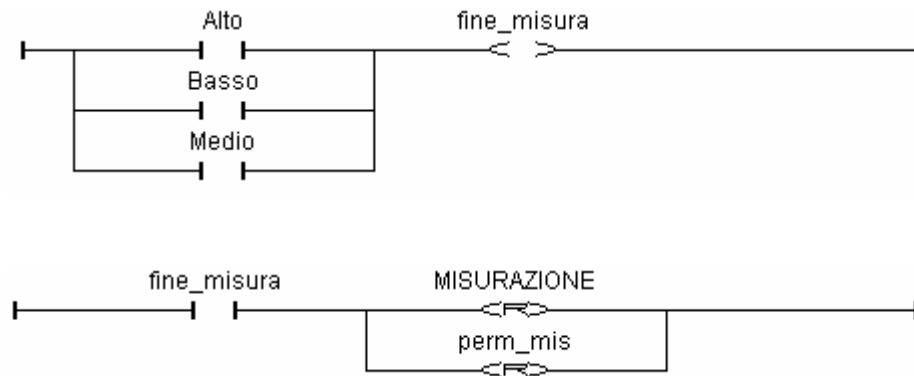



fig. 6 *misur*

➢ 2.2.5 Select

Nel momento in cui la variabile *acceso* è vera, il programma ***super*** ha dato il permesso *perm_selec* ed è stato individuato il tipo di pezzo, si deve accendere il led corrispondente alla terza postazione di lavoro.

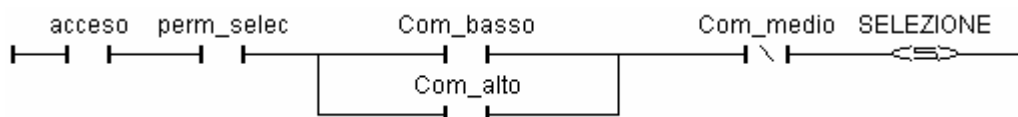



fig.1 *select*

Settata la variabile *SELEZIONE* va dato il comando *sel_BASSO (ruota verso destra)* o *sel_ALTO (ruota verso sinistra)* a seconda del tipo di pezzo che è stato misurato .

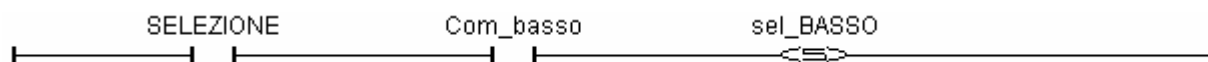



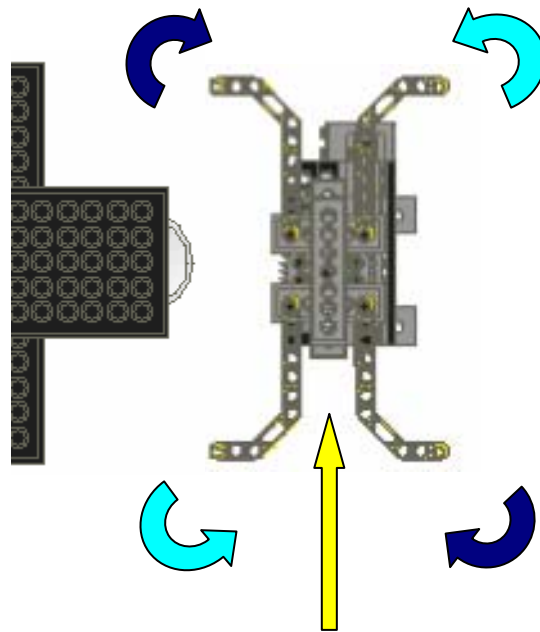
fig. 2 *select*

Nel precedente caso è stato misurato un pezzo basso e quindi si da il comando al motore della postazione di selezione di ruotare verso destra.

SELEZIONE Com_alto sel_ALTO <S>

fig. 3 *select*

Se invece è stato misurato un pezzo alto si richiede al motore di ruotare in verso opposto in modo da mantenere separati i pezzi scartati (solo i pezzi di mezza altezza sono ritenuti adatti).

Dalle fig. 2 e fig. 3 si ricava ancora una volta l'importanza che ha per il controllo la variabile *acceso* grazie alla quale si riesce a settare *SELEZIONE* da cui dipendono i comandi che determinano lo svolgimento dell'operazione di selezione.

Il principio da usare per riconoscere la fine della rotazione nell'operazione di selezione è lo stesso già esposto a pag. 13 . Si deve riuscire a riconoscere il fronte di salita dello switch (*sw_sel*) posto a lato della postazione (indicata dalla freccia gialla nella figura precedente).

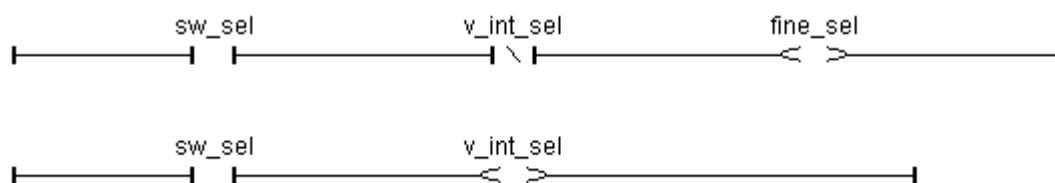



fig. 4 *select*

Attivata la variabile *fine_selezione* viene spento il led corrispondente alla stazione di lavoro e resettati sia i comandi di selezione che le variabili che determinano il comportamento del sistema, nonché il permesso corrispondente.

Infatti una volta selezionato il pezzo, alto o basso che sia, il sistema ha compiuto l'operazione dettatagli dalle variabili *Com_"x"* , la loro influenza sul suo comportamento è terminata.

fig. 5 *select*

➢ 2.2.6 Unload

La logica da usare per implementare il programma della stazione di scarico è la stessa discussa nel paragrafo precedente.
Quando il sistema è in uno stato di normale funzionamento, ovvero il programma ***super*** ha dato il permesso di scaricare e il pezzo identificato come medio è arrivato davanti alla postazione, si deve accendere il led *SCARICO* ad indicare l'inizio dell'operazione.

fig. 1 *unload*

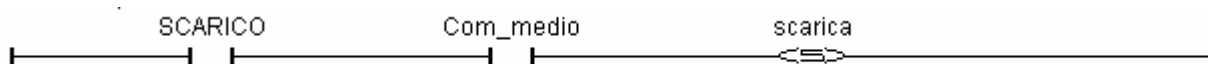



fig. 2 *unload*

Osservando la fig. 2 si capisce che il comando *scarica* è anch'esso legato allo stato della variabile *acceso*.

Anche l'operazione di scarico avviene grazie alla rotazione di braccia lego che tolgono il pezzo dalla tavola rotante.
In modo del tutto analogo a quanto visto per la selezione si deve dunque riconoscere il fronte di salita dello switch *sw_scarico* per fermare l'operazione.

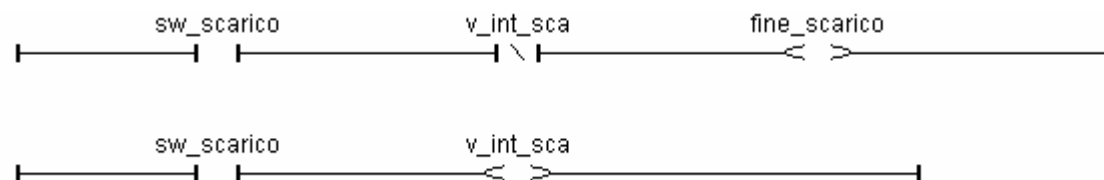



fig. 3 *unload*

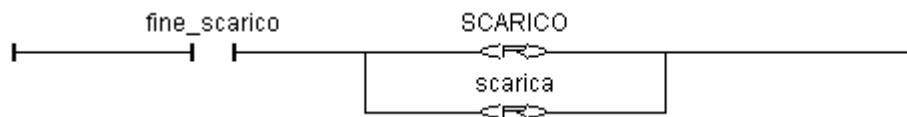



fig. 4 *unload*

Terminata la rotazione lo scarico del pezzo è completato; si deve settare la variabile *fine_scarico* e tramite questa operare sul numero dei pezzi medi presenti nel sistema.
In questa fase si devono implementare le ultime righe di pseudocodice di pag. 16 relative alla gestione dei pezzi di media altezza.

Quando un pezzo medio viene scaricato dalla tavola rotante la variabile *num_medi* deve essere decrementata di un'unita, operazione compiuta tramite un blocco sottrattore:

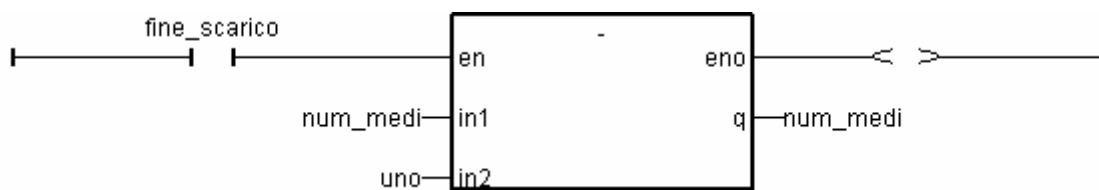



fig. 5 *unload*

Mediante un blocco di confronto si riesce poi a stabilire quando non vi sono più presenti pezzi medi in alcuna postazione del sistema e di conseguenza va resettata la variabile *Com_medio* :

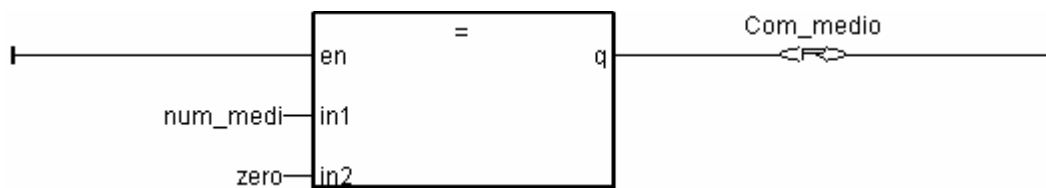



fig. 6 *unload*

➢ 2.2.7 DANGER

Nella gestione di un impianto industriale reale è necessario prevedere un comando in grado di bloccare all'istante ogni lavorazione in corso.
Questo ruolo è affidato alla variabile interna *emergency* settata, insieme al led PERICOLO, qualora si verificasse un'emergenza, attivando l'omonimo ingresso.

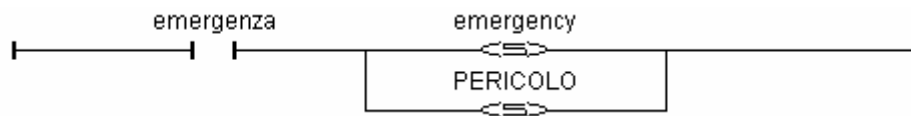



fig. 1 *DANGER*

Per essere in grado di fermare istantaneamente tutte le operazioni in corso si devono resettare tutti i comandi impartiti dettata dal fatto che le diverse lavorazioni possono arrivare al termine solo se per tutta la durata dell'operazione il comando che le attiva rimane vero.

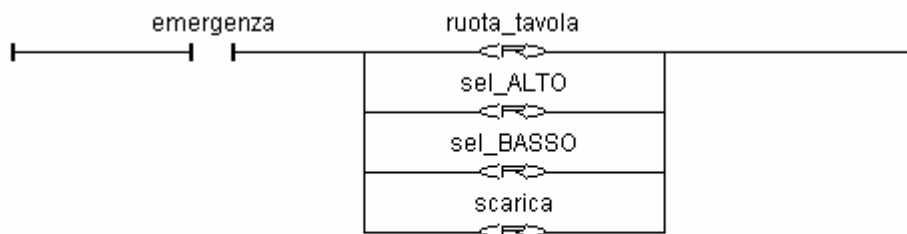



fig. 2 *DANGER*

➢ 2.2.8 Errori

Nel controllo di un sistema reale anche la gestione degli errori riveste un'estrema importanza.
Può accadere infatti che una lavorazione non arrivi al termine non solo per via di un'emergenza, ma più semplicemente per qualche malfunzionamento dei sensori piuttosto che dei motori o di qualsiasi altro componente.
In questo caso è bene segnalare l'errore e ancora meglio poter capire quale parte del sistema non funziona correttamente.

Un metodo semplice per riconoscere uno stato di malfunzionamento è l'assegnazione di un timeout per ogni operazione al fine di indicare il tempo massimo entro cui la lavorazione va eseguita.
Trascorso questo tempo se il processo non è ancora terminato bisogna segnalare l'errore.
In questo modo è anche molto semplice riuscire ad individuare quale operazione non si è conclusa correttamente.

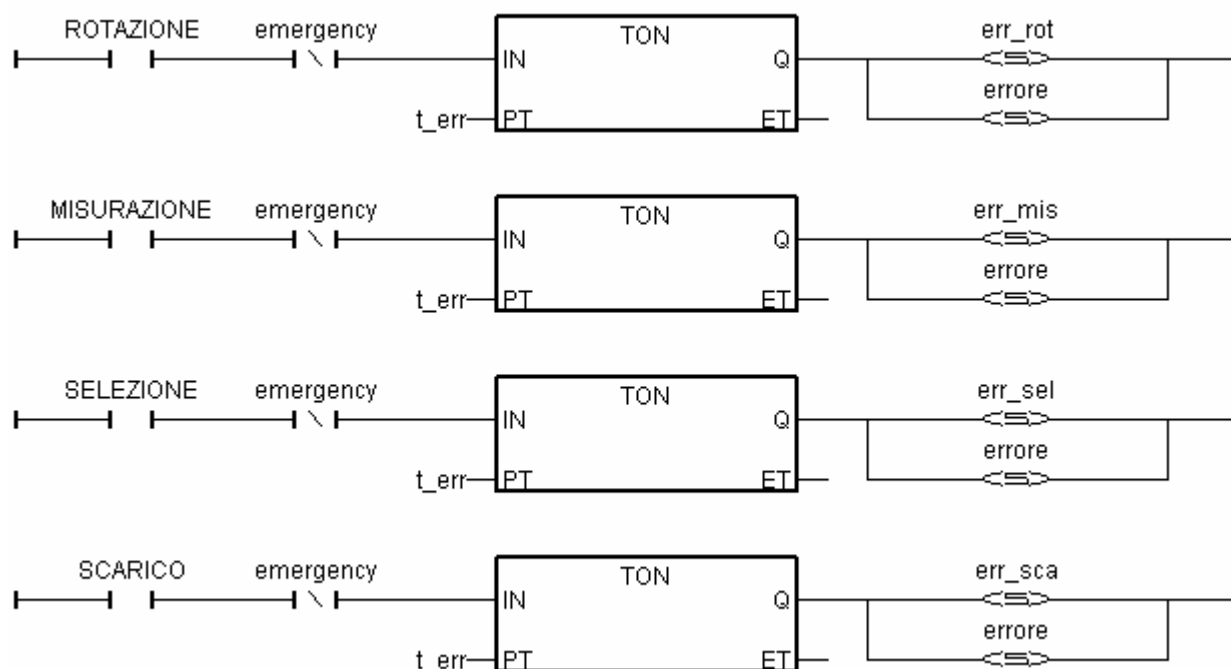



fig. 1 *errori*

Anche in questo caso si deve utilizzare un blocco TON per poter confrontare il tempo trascorso con il tempo massimo impostato, il timeout.

La variabile *t_err* è inizializzato a 6 secondi, valore che si può tranquillamente mantenere per tutte le operazioni.

Nella gestione degli errori tramite timeout si deve tener presente il ruolo della variabile interna *emergency* : quando l'intero sistema viene fermato il conteggio del contatore del blocco TON continua e una volta superato il valore di *t_err* viene segnalato l'errore.

Per evitare ciò bisogna settare la variabile *errore* solo nel caso in cui il sistema è in uno stato di normale funzionamento. Ciò si ottiene ponendo il contatto negato *emergency* prima del blocco TON.

Al pari del verificarsi di un'emergenza improvvisa, se si verifica un errore si devono resettare tutti i comandi che permettono lo svolgimento delle diverse operazioni del sistema .

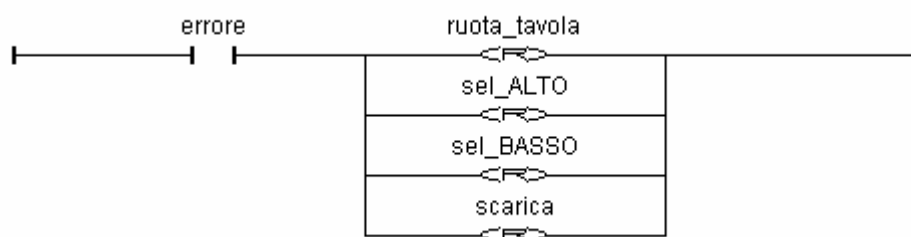



fig. 2 ***errori***

La segnalazione dell'errore deve rimanere attiva sino allo spegnimento del sistema.

Quando non si ha più la condizione di *start* (cioè il sistema viene spento) bisogna resettare le uscite corrispondenti agli errori.

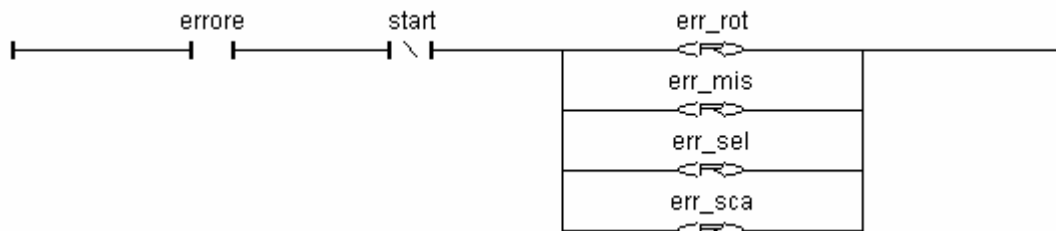



fig. 3 ***errori***

2.3 ANALISI DEL SIMULATORE

Per poter provare l'affidabilità del supervisore si deve implementare un simulatore che consenta inoltre di testare i casi di malfunzionamento.
La scelta del linguaggio da utilizzare è libera in quanto il simulatore non dovrà essere riscritto in KOP.
Il linguaggio più opportuno è il Sequential Function Chart (SFC) che permette di progettare in maniera rapida e tutto sommato semplice un buon simulatore.

Un'importante annotazione riguarda le variabili intere d'ingresso simSET, simROT, simSCA, simSEL con le quali un operatore esterno può variare l'esecuzione della simulazione seguendo la casistica descritta nelle seguenti tabelle.

simSET	DESCRIZIONE
0	Condizioni iniziali rispettate
1	comando *ruota_tavola* inizialmente attivo
2	comando *sel_BASSO* inizialmente attivo
3	comando *sel_ALTO* inizialmente attivo
4	comando *scarica* inizialmente attivo
5	sensore *sw_r* inizialmente inattivo
6	sensore *sw_sel* inizialmente inattivo
7	sensore *sw_sca* inizialmente inattivo
8	sensore *sens_luce1* inizialmente attivo
9	sensore *sens_luce2* inizialmente attivo
10	sensore *sens_luce3* inizialmente attivo
11	sensore *sens_luce4* inizialmente attivo

Tabella 2.1

simROT	DESCRIZIONE
0	Condizioni di normale funzionamento
1	Timeout di rotazione non rispettato

Tabella 2.2

simSEL	DESCRIZIONE
0	Condizioni di normale funzionamento
1	Timeout di selezione non rispettato

Tabella 2.3

simSCA	DESCRIZIONE
0	Condizioni di normale funzionamento
1	Timeout di scarico non rispettato

Tabella 2.4

Un'altra importante considerazione va fatta per le variabili temporali con le quali un operatore può variare il tempo esecutivo delle diverse operazioni.
Per fare ciò si introducono 4 variabili intere d'ingresso *t1, t3, t4* rispettivamente per la rotazione, la selezione e lo scarico tramutate nelle rispettive variabili temporali interne *temp1, temp3, temp4* tramite le seguenti **istruzioni ST** introdotte in ogni fase iniziale del corrispettivo programma di simulazione.

```
ACTION (N) :
        temp x:=tmr(t x*1000);
END_ACTION;
```

LISTA DELLE VARIABILI DEI PROGRAMMI DI SIMULAZIONE

→ PROGRAMMA *ACCENSIO*

- **start** : variabile d'ingresso con cui accendere/spegnere il sistema
- **sw_r** : switch posto nella postazione di carico che serve a rilevare la rotazione della tavola
- **sw_select** : sensore che indica la posizione del selezionatore
- **sw_scarico** : sensore che indica la posizione dello scaricatore
- **scarica** : variabile interna che da il comando di rotazione allo scaricatore
- **sel_BASSO** : variabile interna che da il comando di rotazione verso destra al selettore per selezionare il pezzo Basso
- **sel_ALTO** : variabile interna che da il comando di rotazione verso sinistra al selettore per selezionare il pezzo Alto
- **ruota_tavola**: variabile interna che da il comando di rotazione alla tavola
- **sens_luce1** : sensore che rileva la presenza del pezzo in posizione 1
- **sens_luce2** : sensore che rileva la presenza del pezzo in posizione 2
- **sens_luce3** : sensore che rileva la presenza del pezzo in posizione 3
- **sens_luce4** : sensore che rileva la presenza del pezzo in posizione 4
- **simSET** : variabile intera d'ingresso per simulare il setup della macchina

→ PROGRAMMA *CARICO*

- **start** : variabile d'ingresso con cui accendere/spegnere il sistema
- **pezzoAlto**: variabile d'ingresso con cui simulare il carico di un pezzo Alto
- **pezzoBasso**: variabile d'ingresso con cui simulare il carico di un pezzo Basso
- **pezzoMedio**: variabile d'ingresso con cui simulare il carico di un pezzo Medio
- **sens_luce1** : sensore che rileva la presenza del pezzo in posizione 1
- **A** : variabile interna con cui il programma comunica alla stazione di misura che i il pezzo caricato è alto
- **B** : variabile interna con cui il programma comunica alla stazione di misura che il pezzo caricato è alto
- **S** : variabile interna con cui il programma comunica alla stazione di misura che il pezzo caricato è alto

→ PROGRAMMA *SIM1- SIM_ROT*

- **start** : variabile d'ingresso con cui accendere/spegnere il sistema
- **perm_rot, perm_rot1** : variabili interne indicanti il permesso di rotazione
- **fine_rotazione** : variabile interna indicante la fine della rotazione
- **sens_luce1** : sensore che rileva la presenza del pezzo in posizione 1
- **sens_luce2** : sensore che rileva la presenza del pezzo in posizione 2
- **sens_luce3** : sensore che rileva la presenza del pezzo in posizione 3
- **sens_luce4** : sensore che rileva la presenza del pezzo in posizione 4
- **sw_r** : switch posto nella postazione di carico che serve a rilevare la rotazione della tavola
- **ruota_tavola**: variabile interna che da il comando di rotazione alla tavola
- **t1** : variabile intera d'ingresso con cui decidere il tempo di esecuzione della rotazione
- **temp1** : variabile temporale interna
- **simROT** : variabile intera d'ingresso per simulare il funzionamento/errore della rotazione

→ PROGRAMMA *SIM2*

- **s1_mis, s2_mis, s3_mis** : sensori luce che misurano l'altezza del pezzo
- **perm_mis**: variabili interna indicante il permesso di misurare
- **fine_mis** : : variabile interna indicante la fine della rotazione
- **A, B, S** : variabili interne dettate dal ***carico***

→ PROGRAMMA *SIM3 – SIM_SEL*

- **start** : variabile d'ingresso con cui accendere/spegnere il sistema
- **sens_1uce3**: sensore che rileva la presenza del pezzo posizione 3
- **perm_select** : variabili interne indicante il permesso di selezione
- **fine_sel** : variabile interna indicante la fine della selezione
- **sw_sel** : sensore che indica la posizione del selezionatore
- **t3** : variabile intera d'ingresso con cui decidere il tempo di esecuzione della selezione
- **temp3** : variabile temporale interna
- **sel_BASSO** : variabile interna che da il comando di rotazione verso destra al

selettore per selezionare il pezzo Basso

- **sel_ALTO** : variabile interna che da il comando di rotazione verso sinistra al selettore per selezionare il pezzo Alto
- **simSEL** : variabile intera d'ingresso per simulare il funzionamento/errore della selezione

→ PROGRAMMA *SIM4 – SIM_SCA*

- **start** : variabile d'ingresso con cui accendere/spegnere il sistema
- **sens_1uce4** : sensore che rileva la presenza del pezzo posizione 4
- **perm_scarico** : variabili interna indicante il permesso di scaricare
- **fine_scarico** : variabile interna indicante la fine dello scarico
- **sw_sca** : sensore che indica la posizione dello scaricatore
- **t4** : variabile intera d'ingresso con cui decidere il tempo di esecuzione dello scarico
- **temp4** : variabile temporale interna
- **simSCA** : variabile intera d'ingresso per simulare il funzionamento/errore dello scarico

➢ 2.3.1 Accensio

Al momento dell'accensione del sistema le uniche variabili vere sono quelle che rappresentano gli switch, mentre lo stato dei comandi, così come quello dei sensori e degli switch deve essere falso.

Quando la variabile *start* è vera e *simSET* vale 0 allora l'accensione avviene senza problemi; al contrario quando *simSET* vale 1, 2, … ,11 viene settata la variabile corrispondente ad un comando, ad un sensore o ad uno switch (come mostrato nella precedente tabella 1).

In questo modo verrà settato dal controllore la variabile d'uscita e*rr_setup,* in quanto le condizioni iniziali nono sono rispettate.

L'unico modo per ovviare a questo errore è spegnere il sistema e riavviarlo.

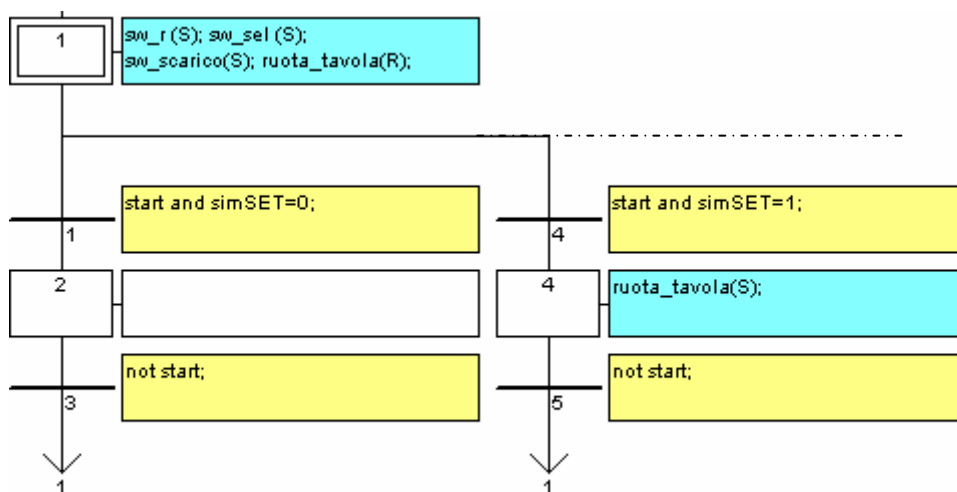



fig. 1 *accensio*

➢ 2.3.2 Carico

La simulazione dell'operazione di carico deve semplicemente settare il sensore luce 1 (*sens_luce1*) ed, in base al tipo di pezzo caricato, una tra le variabili interne *B*, *M* o *A* utilizzate per comunicare con il programma **sim2.**

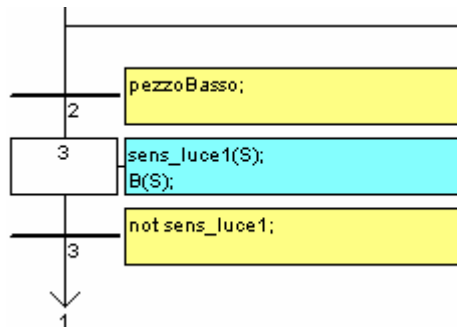



fig. 2 ***carico***

➢ 2.3.3 Sim1 – Sim_Rot

Il programma ***sim1*** consiste principalmente in 6 rami tra loro esclusivi che rappresentano tutti i possibili casi di condizione plausibile per l'esecuzione della rotazione, ovvero:

- macchina accesa (*start*) – permesso di rotazione (*perm_rot*) – *sens_luce1* attivo
- macchina accesa (*start*) – permesso di rotazione (*perm_rot*) – *sens_luce1* attivo – *sens_luce2* attivo
- macchina accesa (*start*) – permesso di rotazione (*perm_rot*) – *sens_luce1* attivo – *sens_luce2* attivo – *sens_luce3* attivo
- macchina accesa (*start*) – permesso di rotazione (*perm_rot*) – *sens_luce3* attivo – *sens_luce3* attivo
- macchina accesa (*start*) – permesso di rotazione (*perm_rot*) – *sens_luce2* attivo
- macchina accesa (*start*) – permesso di rotazione (*perm_rot1*) – *sens_luce3* attivo

Il programma **sim1** chiama il proprio figlio ***sim_rot*** tramite le fasi 2, 4, 6, 8, 10, 12 ed aspetta l'attivazione della variabile *fine_rotazione* per settare i nuovi sensori a seconda della situazione di partenza.

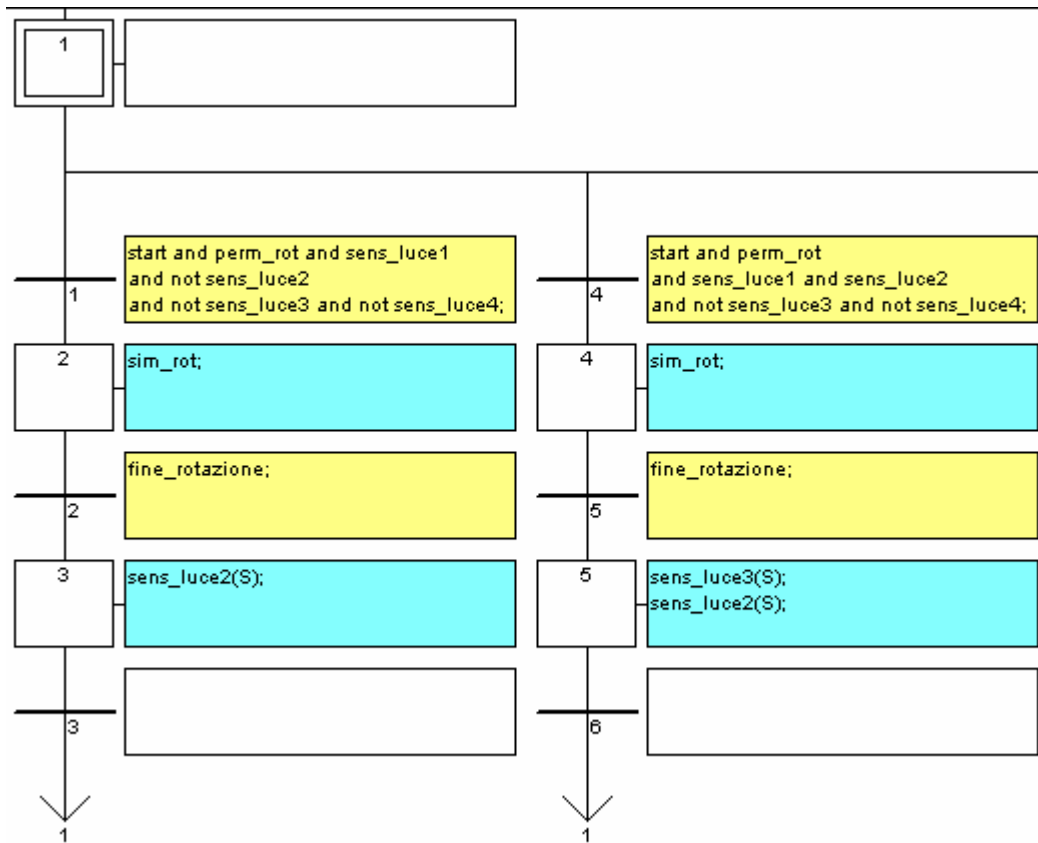



fig. 3 ***sim1*** - *rami esemplificativi* -

In fig. 4 *rot* è rappresentato il programma ***sim_rot*** nel quale la prima fase serve a riconosce il tempo necessario ad eseguire la rotazione indicato dell'operatore - *t1*-.

La condizione per proseguire è l'attivazione, effettuata dal controllore, del comando *ruota_tavola* e della presenza del *sw_r*.

Dopodiché, nella fase 2 vanno resettati tutti i sensori luce e lo switch di rotazione che dopo il tempo *temp1*, uguale a *t1*, ed in caso di *simROT* = 0 viene settata di nuovo in modo da simulare il fronte di salita dello *sw_r*.

In caso di *simROT* = 1 tutto questo non avviene e di conseguenza il controllore segnalerà *err_rot.*

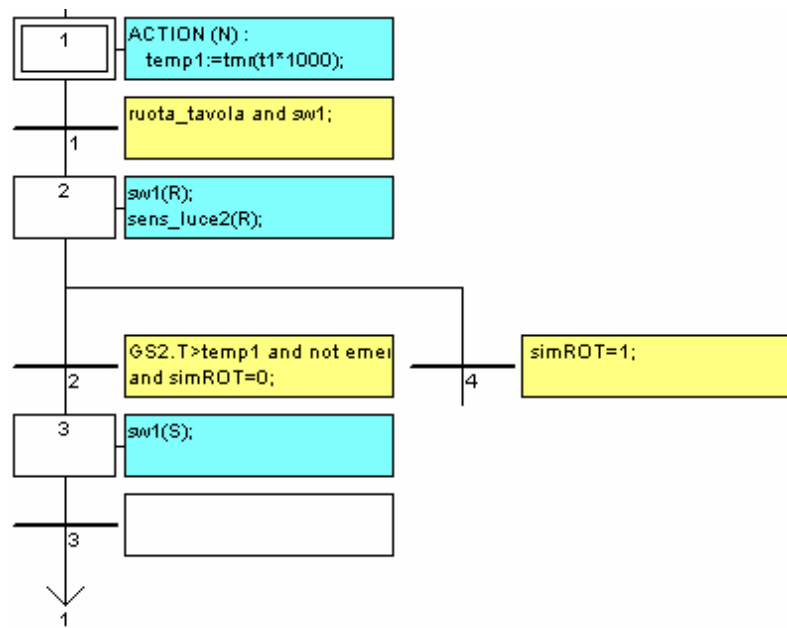



fig. 4 ***rot***

NOTA : per il codice dell'intero programma si rimanda all'*APPENDICE B*

➢ 2.3.4 Sim2

In sostanza il programma ***sim2*** simula il settaggio delle variabili *s1_mis, s2_mis, s3_mis* a seconda del tipo di pezzo misurato e termina, ritornando nella situazione iniziale, quando si attiva il la variabile *fine_misura* come esemplificato in fig. 6 *sim2.*

NOTA: si ricorda che *B, A, M* sono variabili interne settate in ***carico*** quando si inserisce rispettivamente un *pezzoBasso*, un *pezzoAlto*, un *pezzoMedio*.

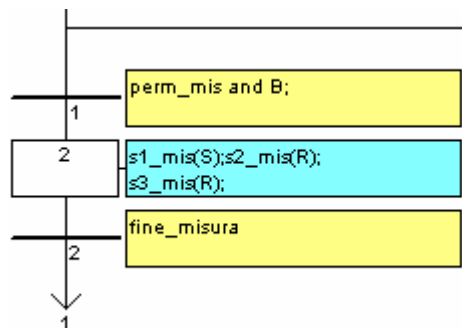



fig. 5 *sim2*

NOTA : per il codice dell'intero programma si rimanda all'*APPENDICE B.*

➢ 2.3.5 Sim3 – Sim_sel

Il programma di simulazione della selezione funziona con la medesima logica implementativa di quello per la rotazione spiegato nel paragrafo 2.2.3, ovvero quando la condizione necessaria è vera, il programma ***sim3*** chiama il proprio figlio ***sim_sel*** che preseguirà solamente se riceve dal controllore il comando *sel_ALTO* o *sel_BASSO*, e se lo switch *sw_sel* è attivo.
Di conseguenza quest'ultimo va resettato e poi settato una volta che sono trascorsi *temp3* secondi indicati dall'operatore mediante la variabile intera *t3*.

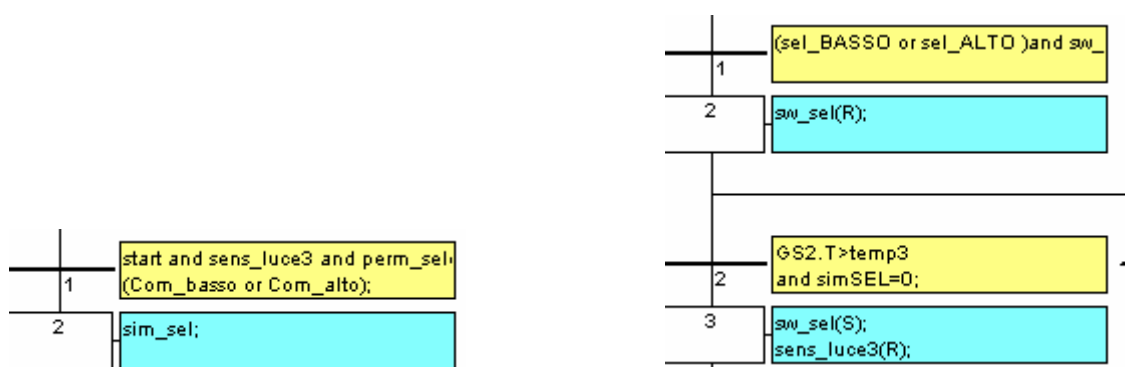



fig.6 *sim3-sim_sel*

NOTA : per il codice dell'intero programma si rimanda *all'APPENDICE B*

2.3.6 Sim4 – Sim_sca

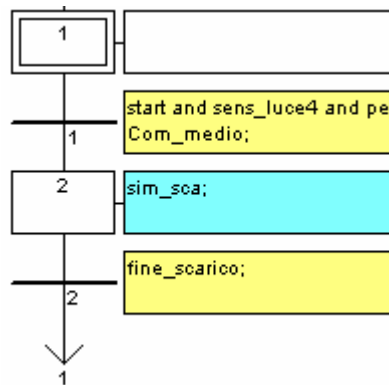



fig. 7 *sim4*

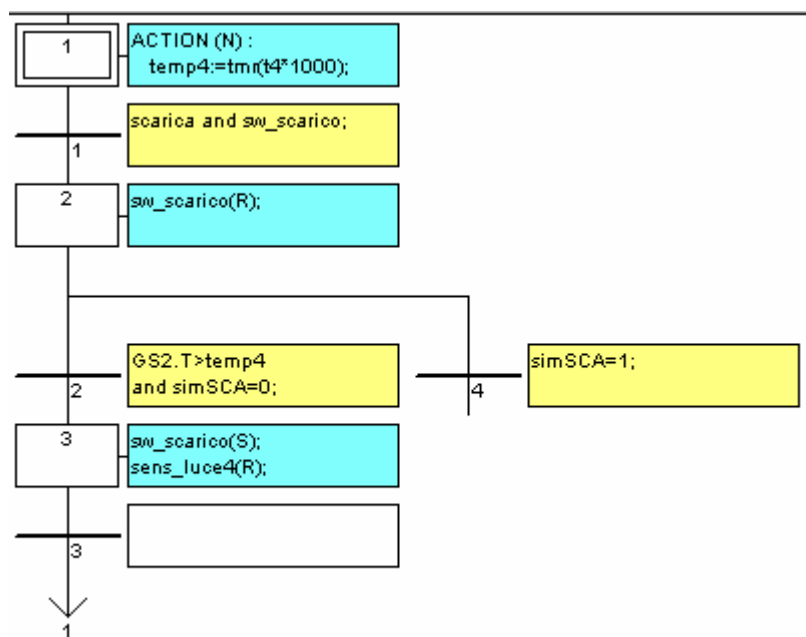



fig. 8 *sim_sca*

Le figure precedenti rappresentano la simulazione della stazione di scarico la quale, come si può notare, segue la stessa logica delle altre operazioni.

In questo caso il rispettivo comando è *scarica*, lo switch è *sw_scarico*, le variabili timer *temp4* e *t4* ed infine la variabile intera è *simSCA*.

2.4 DEBUGGER GRAFICO

I programmi graph_*B, graph_A e graph_M* sono necessari per visualizzare graficamente durante l'esecuzione la posizione del pezzo sulla tavola rotante (rispettivamente del pezzo Basso, Alto e Medio).
Per fare ciò sono state create delle variabili ausiliarie per ogni posizione e per ogni tipo di pezzo riassunte nella seguente tabella.

basso_pos1	Attiva se un pezzo basso si trova in posizione 1
basso_pos2	Attiva se un pezzo basso si trova in posizione 2
basso_po3	Attiva se un pezzo basso si trova in posizione 3
alto_pos1	Attiva se un pezzo alto si trova in posizione 1
alto_pos2	Attiva se un pezzo alto si trova in posizione 2
alto_pos3	Attiva se un pezzo alto si trova in posizione 3
medio_pos1	Attiva se un pezzo medio si trova in posizione 1
medio_pos2	Attiva se un pezzo medio si trova in posizione 2
medio_pos3	Attiva se un pezzo medio si trova in posizione 3
medio_pos4	Attiva se un pezzo medio si trova in posizione 4

NOTA: Per il codice dei programmi si rimanda all' *APPENDICE B*.

Immagine tratta dalla simulazione nell'ambiente ISaGRAF.

Capitolo 3

Collegamenti elettrici al PLC Siemens S7-200

3.1 Presentazione del PLC utilizzato

Il PLC utilizzato nell'esperimento è stato fornito dalla SIEMENS.
Appartiene alla serie S7-200 la quale è costituita da controllori programmabili di dimensioni ridotte (microcontrollori) in grado di controllare un'ampia gamma di dispositivi utilizzabili nei più svariati task di automazione.
L'S7-200 controlla gli ingressi e modifica le uscite in base al programma utente il quale può comprendere operazioni booleane, di conteggio, di temporizzazione, operazioni matematiche complesse e funzioni di comunicazione con altri dispositivi intelligenti.
La struttura compatta, la configurazione flessibile e il vasto set di operazioni fanno dei controllori S7-200 una soluzione ottimale per la gestione di un'ampia varietà di applicazioni.

Il primo elemento costituente il PLC è un alimentatore che eroga una tensione di 24 Volt al quale è stato collegato un microcontrollore tipo CPU 222, appartenente appunto alla serie S7-200, rappresentato in figura 1 e con le caratteristiche riassunte nella tabella di pag. 353 e seguenti della documentazione fornita su supporto digitale da SIEMENS.

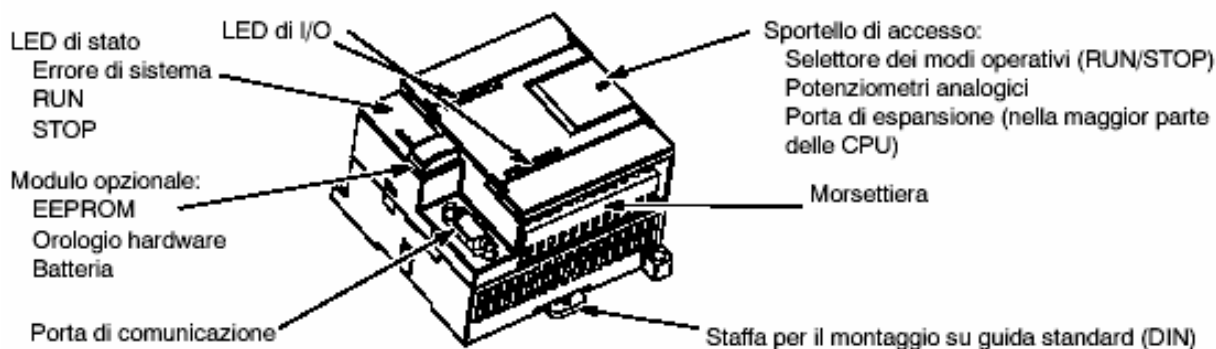



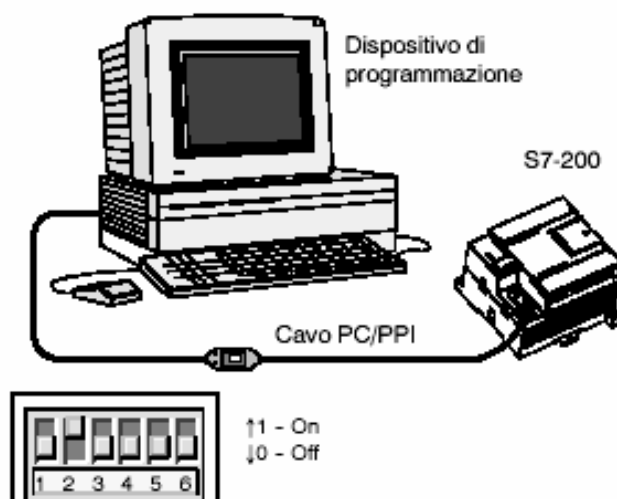
figura 1- CPU 222

Oltre al microcontrollore è stata utilizzata una unità digitale di ampliamento la quale consente di estendere le funzioni della CPU grazie alle caratteristiche riportate nelle tabelle di pag 359 e seguenti.

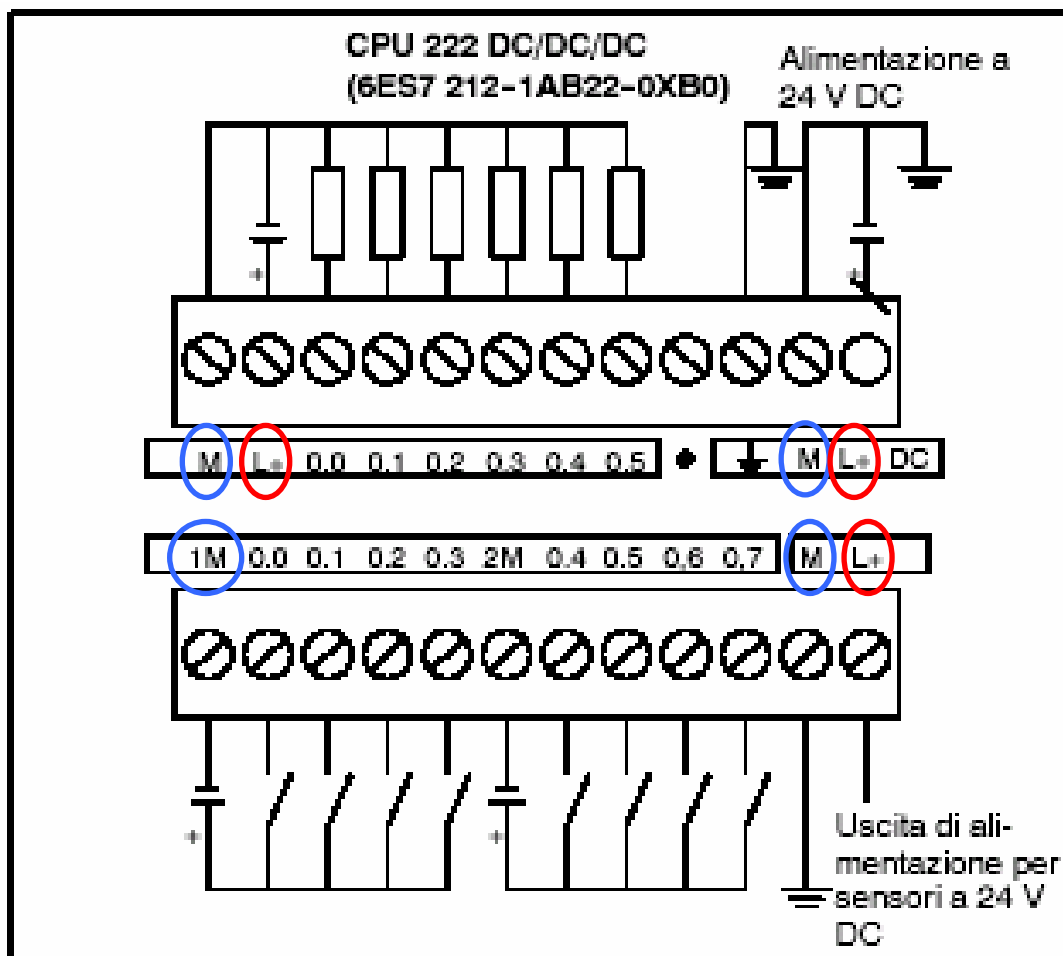
In abbinamento al PLC viene naturalmente fornito il pacchetto di programmazione STEP 7--Micro/WIN il quale mette a disposizione un ambiente di facile utilizzo per lo sviluppo, la modifica e la supervisione della logica necessaria per il controllo di un'applicazione. I tre editor in dotazione consentono di sviluppare il programma di controllo in modo pratico ed efficiente. Inoltre, per facilitare il reperimento delle informazioni, STEP 7--Micro/WIN viene fornito con un'esaustiva guida in linea e un CD di documentazione ai quali si rimanda per ulteriori informazioni.

➢ 3.1.1 Collegamento del cavo PC/PPI

La figura mostra una CPU S7-200 collegata ad un dispositivo di programmazione per mezzo di un cavo PC/PPI. Per collegare il cavo PC/PPI procedere come indicato di seguito:

1. Inserire il connettore RS-232 del cavo PC/PPI (contrassegnato con la sigla "PC") nella porta di comunicazione del PC.
2. Inserire il connettore RS-485 del cavo PC/PPI (contrassegnato con la sigla "PPI") nella porta 0 o 1 dell'S7-200.
3. Verificare che i DIP switch del cavo PC/PPI siano impostati come indicato nella figura sottostante.

3.2 Collegamenti elettrici

➢ 3.2.1 Alimentazione della CPU e delle Unità digitali

La prima operazione da compiere è quella di alimentare la CPU e l'unita digitale facendo gli opportuni collegamenti, ovvero portando il polo negativo (-) dell'alimentatore in ogni punto contraddistinto dal simbolo M (massa) ed il polo positivo (+) in ogni punto contraddistinto dal simbolo L (load) ad eccezione di quelli sulla morsettiera dell'uscite a Relè (unità digitale) a quali va invece portato il polo positivo (+) dell'alimentatore esterno, al voltaggio con cui si vuole far funzionare i motori LEGO.

collegamenti del polo positivo (+24 V)

collegamenti del polo negativo (massa)

collegamenti del polo positivo (+24 V)

collegamenti del polo negativo (massa)

collegamenti polo positivo dell'alimentazione esterna

➢ 3.2.2 Come collegare un deviatore ad un ingresso del PLC

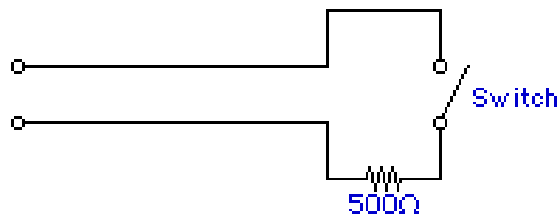
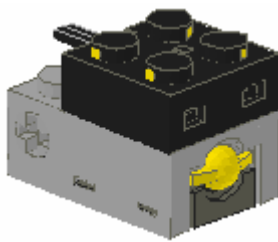
Un deviatore si presenta come in figura dove si può notare la presenza di un contatto Vcc, uno GND e uno C a cui si devono collegare rispettivamente il polo positivo (+), il polo negativo (-) dell'alimentatore e l'ingresso corrispondente del PLC.

➢ 3.2.3 Come collegare un interruttore ad un ingresso del PLC

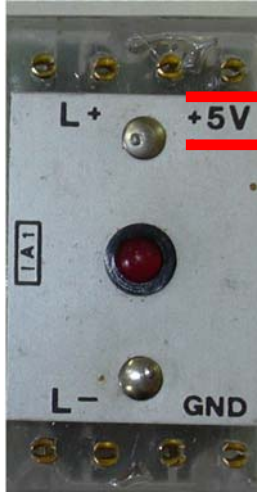
Un interruttore, come si nota in figura, può essere normalmente aperto (NO) oppure normalmente chiuso (NC), in entrambi i casi comunque si deve collegare al contato NO/NC il polo positivo (+) dell'alimentatore, mentre al contatto C va collegato l'ingresso del PLC corrispondente.

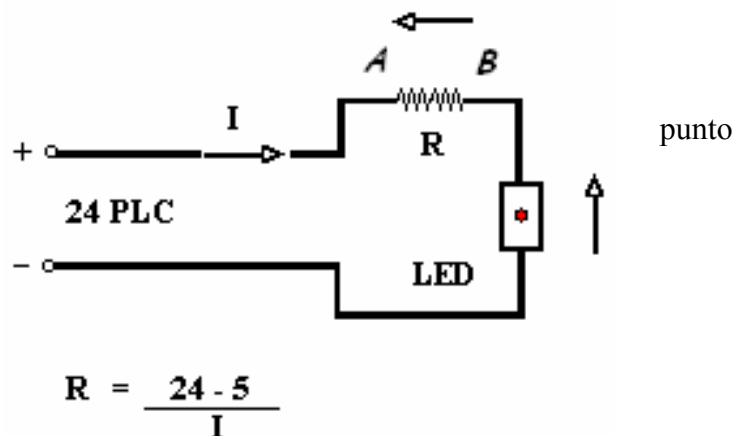
➢ 3.2.4 Come collegare un touch sensor LEGO (switch) ad un ingresso del PLC

Dallo schema elettrico del Touch Sensor LEGO, rappresentato in figura , si deduce che il suo funzionamento è similare a quello di un interruttore normalmente aperto, bisogna dunque collegare a due suoi contati rispettivamente il polo positivo (+) dell'alimentatore e l'ingresso del PLC corrispondente.

Per fare questo si utilizza l'Electric Brick 2 x 2 x 2/3 with Wire End, incastrato sul sensore, portando una estremità del suo filo a contatto con il polo positivo (+) dell'alimentatore e l'altra estremità all'ingresso PLC usando in entrambi i casi, per ragioni cautelari, un "ponte elettrico" ovvero non facendo contatto direttamente con i punti da raggiungere ma tramite un filo elettrico più adeguato.

➢ 3.2.5 Come collegare un light sensor LEGO ad un ingresso del PLC

NOTA: Per quanto riguarda i sensori luce, non sapendo come interfacciarli all'ingresso del PLC, si è deciso di utilizzare dei classici interruttori a cui si collega il polo positivo del PLC al contatto comune ed l'ingresso corrispondente ad uno dei due altri contatti.

➢ 3.2.6 Come collegare un led ad una uscita del PLC

Come si nota dalla figura a lato il LED va alimentato a +5 V di conseguenza, essendo l'alimentazione del PLC a + 24 V, bisogna inserire un resistenza come descritto nello schema sottostante.

Ora al punto A della resistenza va collegata l'uscita del PLC mentre al punto B il contatto + 5 V del LED.
Infine al contatto L – del LED va collegato il polo negativo (-) del PLC.

➢ 3.2.7 Come collegare un motore LEGO ad un'uscita del PLC

Per collegare un motore LEGO si usa un doppio Electric Brick 2 x 2 x 2/3 with Wire End in cui i due Brick sono collegati all'estremità opposte del filo elettrico. Uno dei due Brick va opportunamente incastrato al motore LEGO come descritto nelle ISTRUZIONI GRAFICHE del Capitolo1, mentre all'altro Brick vanno collegati in un polo la massa (-) dell'alimentatore esterno al voltaggio con cui si vuole far funzionare il motore stesso ed all'altro polo l'uscita del PLC (+) che lo comanda.

Capitolo 4

Confronto tra *Ladder Program* e KOP

I due programmi, Ladder Program (LD) dell'ambiente ISaGRAF e KOP di proprietà Siemens, sono molto simili fra loro, entrambi infatti utilizzano una logica booleana basata su contatti e bobine per rappresentare rispettivamente gli ingressi e le uscite di un controllore a logica programmabile. Inoltre mettono a disposizione dell'utente diverse funzioni che permettono di eseguire svariati task di operazioni, da quelle matematiche a quelle di confronto, nonché controlli temporali.

La prima differenza riguarda la dichiarazione delle variabili : nel primo linguaggio è sufficiente dichiarare il nome della variabile, il tipo (booleana, intera/reale, timer, …) e l'attributo corrispondente (interna, uscita, ingresso), mentre nel secondo si deve creare una ***Tabella dei simboli*** in cui definire il nome ed associarlo ad un indirizzo fisico del PLC come raffigurato nelle tabelle: Tabella 4.1, Tabella 4.2 e Tabella 4.3.

Simbolo	**Indirizzo**
start	I0.0
emergenza	I0.1
pezzoBasso	I0.2
pezzoAlto	I0.3
pezzoMedio	I0.4
sens_luce2	I1.1
sens_luce3	I1.2
sens_luce4	I1.3
sw_r	I1.4
sw_sel	I1.5
sw_scarico	I1.6
acceso	Q0.0
PERICOLO	Q0.1
ROTAZIONE	Q0.2
MISURAZIONE	Q0.3
SELEZIONE	Q0.4
SCARICO	Q0.5

ruota_tavola	Q1.1
seleziona *	Q1.3
scarica	Q1.5
errore	Q1.6
err_set	Q1.7

Tabella 4.1 - *variabili input/output* -

Simbolo	Indirizzo
not_iniziali	M7.0
setup_ok	M7.1
emergency	M4.1
ok	M4.2
ok_r	M4.3
ok_sel	M4.4
ok_sca	M4.5
ok_m	M4.6
perm_rot	M4.7
perm_rot1	M5.1
perm_mis	M5.2
perm_selec	M5.3
perm_scarico	M5.4
Fine_rotazione	M5.5
fine_misura	M5.6
fine_sel	M5.7
fine_scarico	M6.1
Com_alto	M6.2
Com_basso	M2.0
Com_medio	M2.1
v_int_rot	M2.2
v_int_sel	M2.3
v_int_sca	M2.4
sens_luce1	M3.3
s1_mis	M2.5
s2_mis	M2.6
s3_mis	M2.7
Alto	M3.0

Medio	M3.1
Basso	M3.2
num_medi	MW20

Tabella 4.2 - *variabili interne –*

Simbolo	Indirizzo
err_rot	M6.3
err_mis	M6.4
err_sel	M6.5
err_sca	M6.6
err	M6.7

Tabella 4.3 – *variabili interne errori –*

NOTA:

- le variabili d'ingresso hanno come indirizzo I *x.y*
- le variabili d'uscita hanno come indirizzo Q *x.y*
- le variabili interne hanno come indirizzo M *x.y* (è importante notare che esistono variabili interne già predefinite per il linguaggio, dunque bisogna fare attenzione a non utilizzare indirizzi già occupati -).
- le variabili numeriche hanno come indirizzo MW *x.y*

dove :

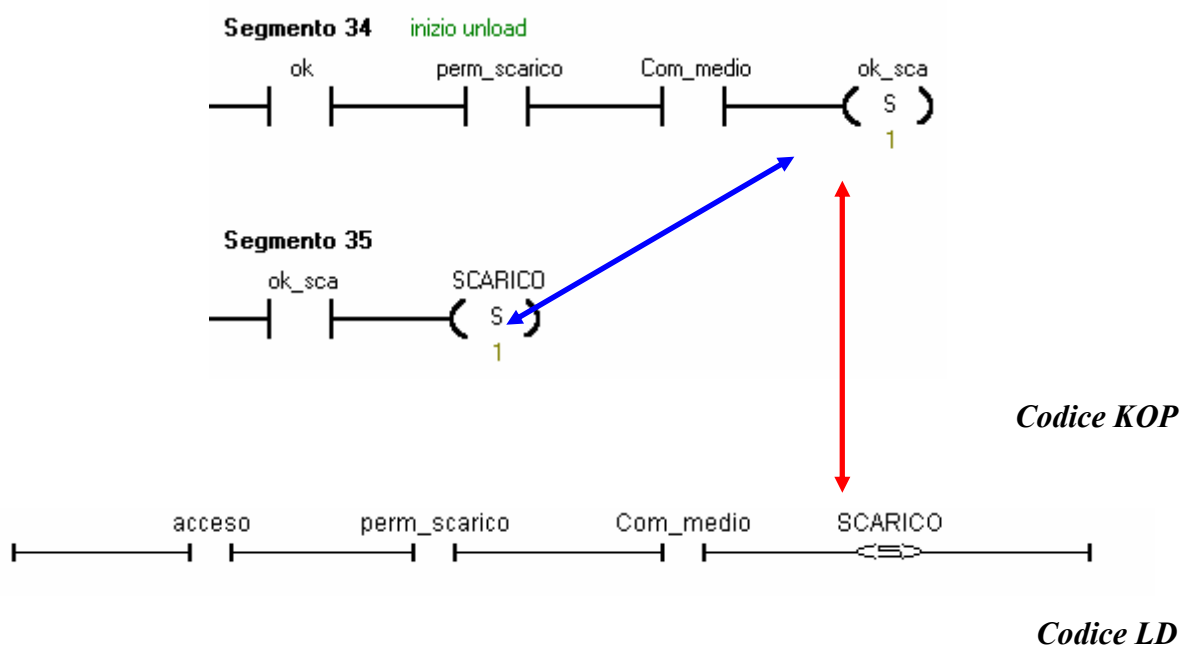
x rappresenta il modulo (0 per la CPU, 1 per l'unità digitale)

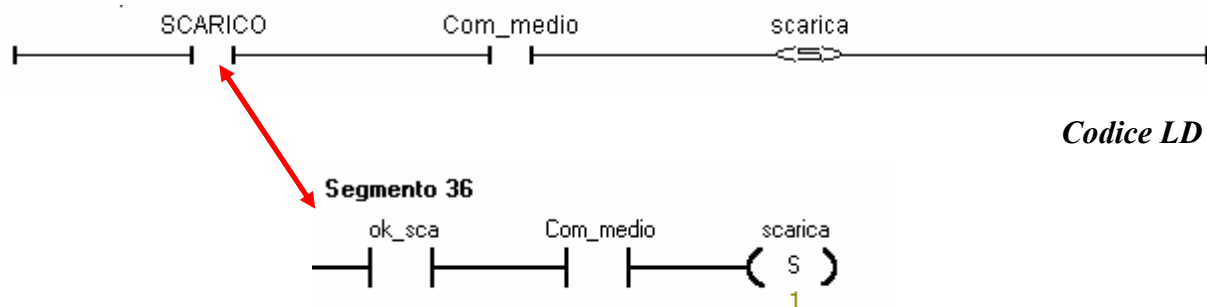
y rappresenta l'ingresso/uscita del modulo (0-7)

* la variabile *seleziona* svolge il compito sia di *sel_BASSO* che *sel_ALTO* (variabili ISaGRAF), ovvero da il comando di ruotare al selezionatore solamente in un verso, di conseguenza i pezzi Alti e Bassi non rimangono separati.

A livello di codice si devono introdurre delle piccole modifiche, di seguito riportate:

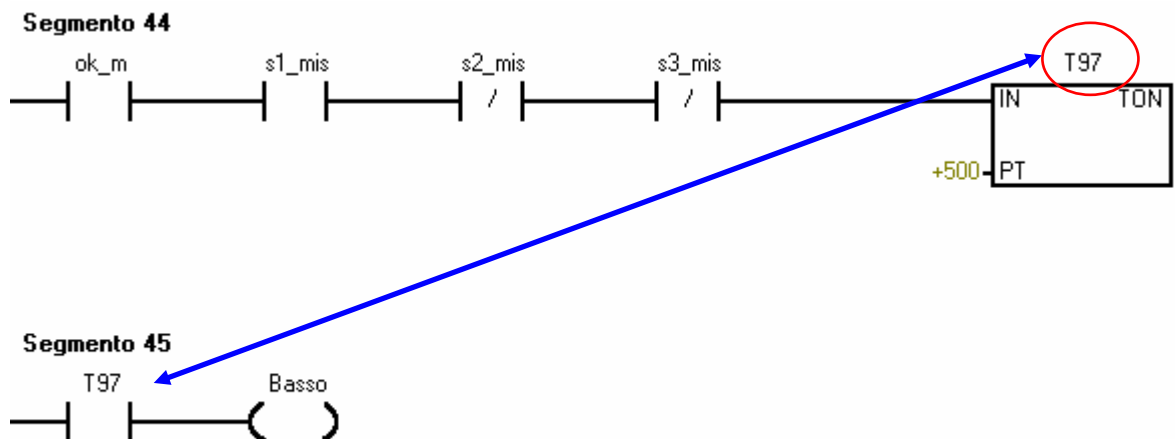
1) Nel KOP le bobine d'uscita vengono settate mediante una variabile interna *ok_"x"* attivata mediante le stesse condizioni della bobina ad essa associata (es. *ok_sca / SCARICO*), ovvero viene posta nella medesima posizione occupata da quest'ultima nel LD.

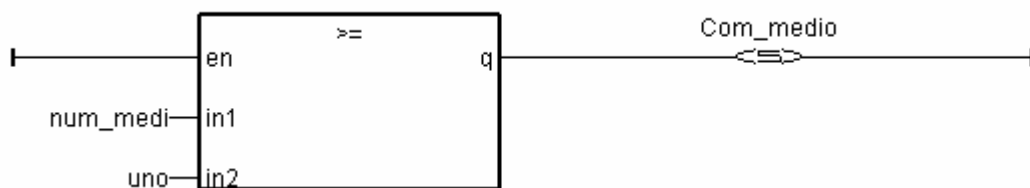
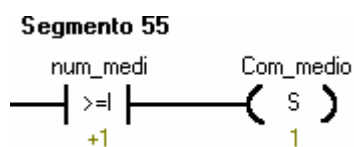
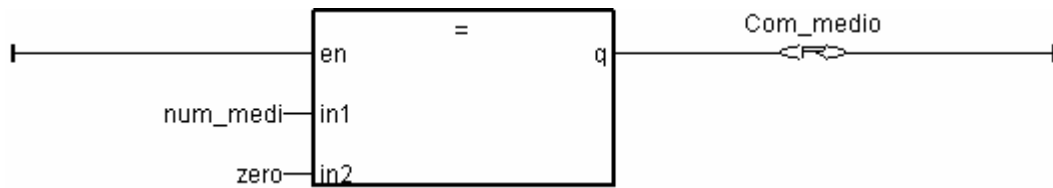
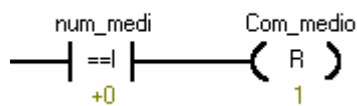
Codice KOP

Codice LD

Ogni qual volta in LD compare il contatto *SCARICO* in KOP viene sostituito con la variabile *ok_sca.*

Codice LD

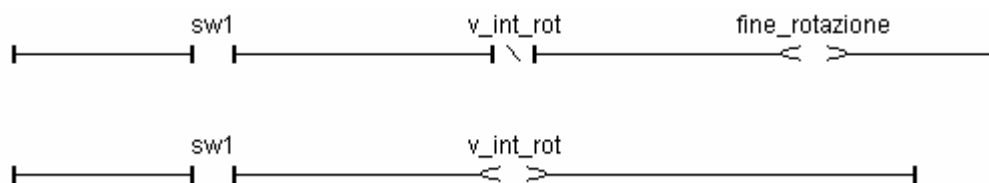
Codice KOP

Questa scelta è dettata dal fatto di non utilizzare variabili d'uscita come contatti, cosicché il controllo avvenga interamente tramite variabili interne.

2) Nel KOP il blocco TON agisce in modo leggermente diverso rispetto ad LD, ad esso è infatti associato un nome predefinito e uno step di conteggio preimpostato.
   E' dunque necessario scegliere il blocco più adatto ed utilizzare il suo nome come contatto nei rami in cui è presente una condizione temporale.
   Per quanto riguarda il funzionamento invece coincide con quello spiegato per il blocco TON del LD nell'*APPENDICE A*.

Codice KOP

3) Nel KOP i blocchi confronto sono sostituiti con contatti appropriati.

Codice LD

Codice KOP

Codice LD

Codice KOP

4) Per riconoscere il fronte di salita in KOP viene usato un particolare contatto.

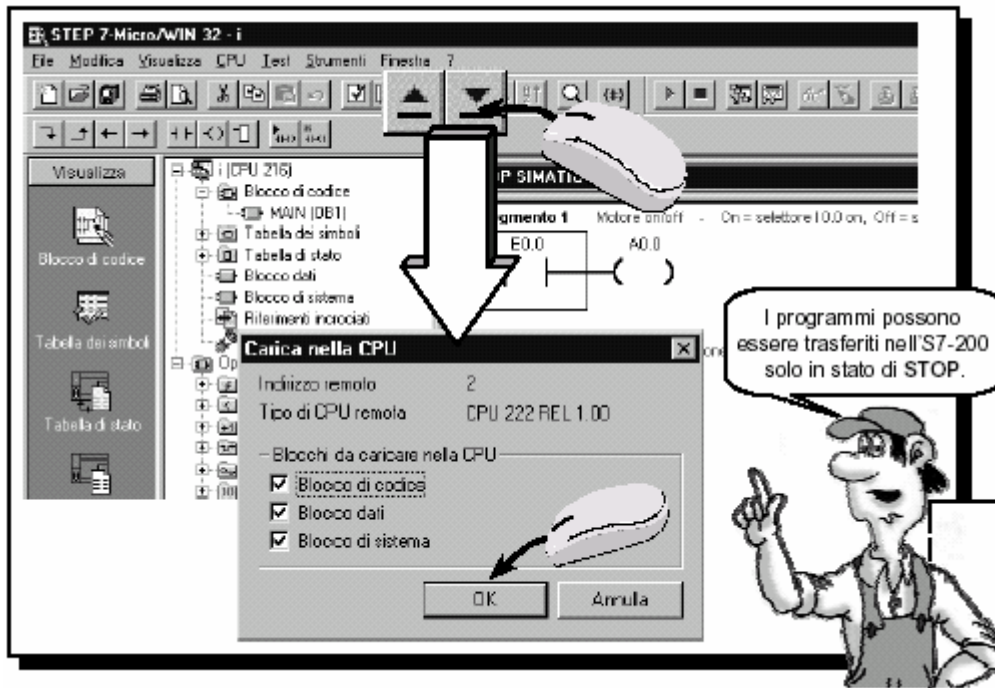
Codice KOP

Codice LD

Mantenendo il metodo usato in ISaGRAF si riscontrano problemi a causa della tempistica del PLC; in alcuni casi la variabile *fine_rotazione* viene attivata anche se non vi è alcun fronte di salita.

NOTA: Il codice KOP completo è riportato nell'*APPENDICE C*.

Caricamento del codice nel PLC

Una volta scritto e compilato il codice va caricato nel PLC eseguendo le seguenti operazioni:

- assicurarsi che il PLC sia in stato di STOP cliccando sull’icona
- cliccare sull’icona

Infine mettere in stato di RUN il PLC, cliccando sull’icona , per eseguire il programma caricato.

Possibile sviluppo e …

Per permettere alla stazione di selezione di dividere i pezzi Alti da quelli Bassi sarebbe necessario che il motore LEGO dedicato possa girare sia in senso orario che antiorario.

Per ottenere ciò si deve invertire la polarità di alimentazione realizzando un circuito elettrico con RELE' a singolo e a doppio scambio.
Bisogna innanzitutto predisporre due uscite (*Qx* e *Qy*) per lo stesso motore, una che dia il permesso di ruotare, l'altra per decidere la direzione.
L'uscita *Qx* eccita la bobina del relè a singolo scambio, mentre *Qy* è collegata al relè a doppio scambio. Quando il primo relè (quello a singolo scambio) conduce, in base allo stato del secondo, si ha la rotazione in un senso o nell'altro.

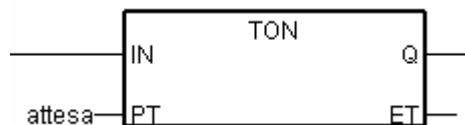
… Conclusioni

L'esperimento è risultato complesso, ma comunque utile per comprendere e verificare quali e quante difficoltà si trovano nel passaggio dalla modellizzazione alla implementazione di un sistema reale, ovvero nel riuscire a combinare la parte di simulazione software con la costruzione fisica del prototipo.

Un ulteriore importante aspetto è stato lo svolgimento del lavoro di ricerca e sviluppo in equipe in modo collaborativi ed efficace.

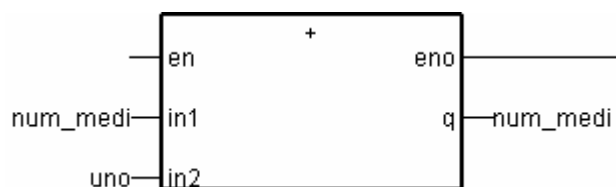
APPENDICE A

Blocchi Funzione

IL BLOCCO "TON" :

IN : variabile booleana ;
se c'è un fronte di salita, inizia a incrementare il timer,
se c'è un fronte di discesa lo ferma e lo azzera.

PT: variabile timer ;
rappresenta il massimo tempo programmato
(*attesa* = 1 secondo)

Q : variabile booleana ;
se è vera il tempo programmato è trascorso

IL BLOCCO SOMMATORE :

in1 : primo input, può essere intero o reale(*num_medi* è inizializzato a 0)

in2 : secondo input, deve essere dello stesso tipo del primo
(*uno* è inizializzato a 1)

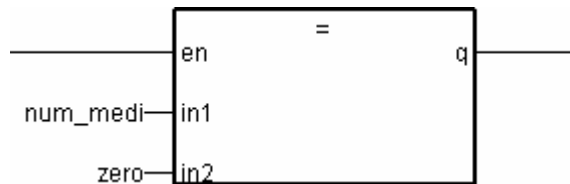
q : output, addizione con segno dei termini in input (è intero o reale a seconda degli input)

IL BLOCCO SOTTRATORE :

in1 : primo input, può essere intero o reale (*num_medi* è inizializzato a 0)

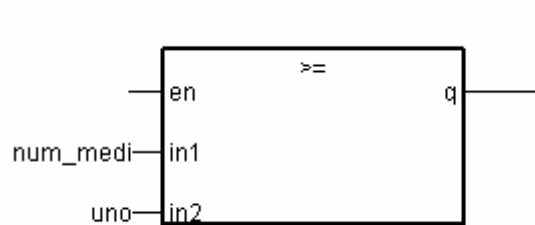
in2 : secondo input, deve essere dello stesso tipo del primo
(*uno* è inizializzato a 1)

q : output, sottrazione del secondo dal primo (è intero o reale a seconda degli input)

IL BLOCCO CONFRONTO :

in1 : primo input, può essere intero, reale o un messaggio (una stringa di caratteri)
(*num_medi* è inizializzato a 0)

in2 : secondo input, deve essere dello stesso tipo del primo
(*uno* è inizializzato a 1)

q : output, è vero se in1 = in2 (è dello stesso tipo degli input)

IL BLOCCO MAGGIORE - UGUALE :

in1 : primo input, può essere intero, reale o un messaggio (una stringa di caratteri)
(*num_medi* è inizializzato a 0)

in2 : secondo input, deve essere dello stesso tipo del primo
(*uno* è inizializzato a 1)

q : output, è vero se in1 >= in2 (è dello stesso tipo degli input)

APPENDICE B

Codice ISaGRAF

Controllore

➢ B.1 Setup

➢ B.2 Super

emergency
start
errore
acceso
acceso
sens_luce1
fine_misura
sw_r
MISURAZIONE
SELEZIONE
SCARICO
perm_rot
acceso Com_medio sens_luce1 sens_luce2 sens_luce3 Com_alto Com_basso sens_luce4 perm_rot1
acceso
sens_luce2
perm_rot
TON
IN
Q
attesa
PT
ET
perm_mis
acceso
sens_luce3
perm_rot
TON
IN
Q
attesa
PT
ET
perm_selec
acceso
sens_luce4
perm_rot
TON
IN
Q
attesa
PT
ET
perm_scarico

➢ B.3 Rotation

acceso
perm_rot
perm_rot1
TON
IN
Q
attesa
PT
ET
ROTAZIONE
perm_mis
perm_selec
perm_scarico

ROTAZIONE
ruota_tavola

sw_r
v_int_rot
fine_rotazione
sw_r
v_int_rot

fine_rotazione
ROTAZIONE
perm_rot
perm_rot1

➢ B.4 Misur

acceso
perm_mis
MISURAZIONE
MISURAZIONE
s1_mis
s2_mis
s3_mis
TON
IN
Q
t_mis
PT
ET
Basso
MISURAZIONE
s1_mis
s2_mis
s3_mis
TON
IN
Q
t_mis
PT
ET
Medio
MISURAZIONE
s1_mis
s2_mis
s3_mis
TON
IN
Q
t_mis
PT
ET
Alto
Basso
Com_basso
Alto
Com_alto
Medio
+
en
eno
num_medi
in1
q
num_medi
uno
in2
>=
en
q
Com_medio
num_medi
in1
uno
in2

Alto
fine_misura
Basso
Medio

fine_misura
MISURAZIONE
R
perm_mis
R

➢ B.5 Select

acceso
perm_selec
Com_basso
Com_medio
SELEZIONE
Com_alto
SELEZIONE
Com_basso
sel_BASSO
SELEZIONE
Com_alto
sel_ALTO
sw_sel
v_int_sel
fine_sel
sw_sel
v_int_sel
fine_sel
SELEZIONE
sel_BASSO
Com_basso
sel_ALTO
Com_alto
perm_selec

➢ B.6 *Uload*

acceso
perm_scarico
Com_medio
SCARICO
SCARICO
Com_medio
scarica
sw_scarico
v_int_sca
fine_scarico
sw_scarico
v_int_sca
fine_scarico
SCARICO
scarica
fine_scarico
-
en
eno
num_medi
in1
q
num_medi
uno
in2
=
en
q
Com_medio
num_medi
in1
zero
in2

➢ B.7 DANGER

emergenza
emergency
<S>
PERICOLO
<S>
emergenza
ruota_tavola
<R>
sel_ALTO
<R>
sel_BASSO
<R>
scarica
<R>

➢ B.8 Errori

ROTAZIONE
emergency
TON
IN
Q
PT
ET
t_err
err_rot
errore
MISURAZIONE
emergency
TON
t_err
err_mis
errore
SELEZIONE
emergency
TON
t_err
err_sel
errore
SCARICO
emergency
TON
t_err
err_sca
errore
errore
ruota_tavola
sel_ALTO
sel_BASSO
scarica
errore
start
err_rot
err_mis
err_sel
err_sca

Simulatore

➢ B.9 Accensio

start and simSET=6;
14
9
sw_sel(R);
not start;
15
1
start and simSET=7;
16
10
sw_scarico(R);
not start;
17
1
start and simSET=8;
18
11
sens_luce1(S);
not start;
19
1
start and simSET=9;
20
12
sens_luce2(S);
not start;
21
1
start and simSET=10;
22
13
sens_luce3(S);
not start;
23
1
start and simSET=11;
24
14
sens_luce4(S);
not start;
25
1

➢ B.10 Carico

1
start;
1
2
pezzoBasso;
2
pezzoMedio;
4
pezzoAlto;
6
3
sens_luce1(S);
B(S);
4
sens_luce1(S);
M(S);
5
sens_luce1(S);
A(S);
not sens_luce1;
3
not sens_luce1;
5
not sens_luce1;
7
1
1
1

➢ B.11 Sim1 – Sim_rot

start and perm_rot
and not sens_luce1 and sens_luce2
and not sens_luce3 and not sens_luce4;
13
10
sim_rot;
fine_rotazione;
14
11
sens_luce3(S);
15
1
start and perm_rot1
and not sens_luce1 and not sens_luce2
and sens_luce3 and not sens_luce4;
16
12
sim_rot;
fine_rotazione;
17
13
sens_luce4(S);
18
1

1
ACTION (N) :
temp1:=tmr(t1*1000);
ruota_tavola and sw1;
1
2
sw1(R);
sens_luce2(R);
GS2.T>temp1 and not emer
and simROT=0;
2
simROT=1;
4
3
sw1(S);
3
1

➢ B.12 Sim2

1
perm_mis and B;
1
2
s1_mis(S); s2_mis(R);
s3_mis(R); B(R);
fine_misura
2
1
perm_mis and M;
3
4
s1_mis(S); s2_mis(S);
s3_mis(R); M(R);
fine_misura
4
1
perm_mis and A;
5
6
s1_mis(S); s2_mis(S);
s3_mis(S); A(R);
fine_misura
6
1

➢ B.13 Sim3 – Sim_sel

➢ B.14 Sim4 – Sim_sca

Debugger Grafico

➢ B.15 Graph_B

3
basso_pos1;
3
4
inizio_rot;
4
5
basso_pos1(R);
fine_rotazione;
5
6
basso_pos2(S);
6
3
7
basso_pos2;
7
8
inizio_rot;
8
9
basso_pos2(R);
fine_rotazione;
9
10
basso_pos3(S);
10
7
11
basso_pos3;
11
12
fine_sel;
12
13
basso_pos3(R);
13
11

➢ B. Graph_A

1
pezzoAlto;
1
2
alto_pos1(S);
2
1
3
alto_pos1;
3
4
inizio_rot;
4
5
alto_pos1(R);
fine_rotazione;
5
6
alto_pos2(S);
6
3

3
alto_pos1;
3
4
inizio_rot;
4
5
alto_pos1(R);
fine_rotazione;
5
6
alto_pos2(S);
6
3
7
alto_pos2;
7
8
inizio_rot;
8
9
alto_pos2(R);
fine_rotazione;
9
10
alto_pos3(S);
10
7
11
alto_pos3;
11
12
fine_sel;
12
13
alto_pos3(R);
13
11

➢ B. Graph_M

1
pezzoMedio;
1
2
medio_pos1(S);
2
1
3
medio_pos1;
3
4
inizio_rot;
4
5
medio_pos1(R);
fine_rotazione;
5
6
medio_pos2(S);
6
3

7
medio_pos2;
7
8
inizio_rot;
8
9
medio_pos2(R);
fine_rotazione;
9
10
medio_pos3(S);
10
7
11
medio_pos3;
11
12
inizio_rot;
12
13
medio_pos3(R);
fine_rotazione;
13
14
medio_pos4(S);
14
11
15
medio_pos4;
15
16
fine_scarico;
16
17
medio_pos4(R);
17
15

APPENDICE C

Codice KOP

➢ C.1 Setup

Segmento 3
ok
not_iniziali
err_set
S
1
ok
R
1
Segmento 4
4
LBL
Segmento 5
ok
not_iniziali
/
setup_ok
S
1
err_set
R
1
Segmento 6
ok
/
setup_ok
R
1

➢ C.2 Super

Segmento 7
inizio super
emergency
/
start
ok
Segmento 8
ok
acceso
S
1
Segmento 9
ok
/
acceso
R
1
Segmento 10
pezzoBasso
sens_luce1
pezzoMedio
pezzoAlto
Segmento 11
pezzoBasso
s1_mis
S
1
s2_mis
R
1
s3_mis
R
1
Segmento 12
pezzoAlto
s1_mis
S
1
s2_mis
S
1
s3_mis
S
1

Segmento 13
pezzoMedio
s1_mis
S
1
s2_mis
S
1
s3_mis
R
1
Segmento 14
ok
sens_luce1
ok_m
ok_sel
ok_sca
perm_rot
S
1
fine_misura
Segmento 15
ok
sens_luce2
perm_rot
T33
IN
TON
+100
PT
Segmento 16
T33
perm_mis
S
1
Segmento 17
ok
sens_luce3
perm_rot
T34
IN
TON
+100
PT

Segmento 18
T34
perm_selec
S
1
Segmento 19
ok
sens_luce4
perm_rot
T35
IN
TON
+100
PT
Segmento 20
T35
perm_scarico
S
1
Segmento 21
ok
Com_medio
sens_luce1
sens_luce2
sens_luce3
sens_luce4
Com_alto
Com_basso
perm_rot1
S
1

➢ C.3 Rotation

Segmento 22
inizio rotation
ok
perm_rot
T36
IN
TON
perm_rot1
+100
PT
Segmento 23
T36
ok_r
S
1
Segmento 23
T36
ok_r
S
1
Segmento 24
ok_r
ROTAZIONE
S
1
ruota_tavola
S
1
Segmento 25
sw_r
P
fine_rotazione
Segmento 26
fine_rotazione
ruota_tavola
R
1
ok_r
R
1
perm_rot
R
1
ROTAZIONE
R
1
perm_rot1
R
1

➢ C.4 Select

Segmento 28
inizio select
ok
sens_luce3
perm_selec
Com_basso
Com_medio
ok_sel
S
1
Com_alto
Segmento 29
ok_sel
SELEZIONE
S
1
Segmento 30
ok_sel
Com_basso
seleziona
S
1
Com_alto
Segmento 31
sw_sel
P
fine_sel
Segmento 32
fine_sel
seleziona
R
1
SELEZIONE
R
1
ok_sel
R
1
Com_basso
R
1
Com_alto
R
1
perm_selec
R
1

➢ C.5 Unload

Segmento 34
inizio unload
ok
perm_scarico
Com_medio
ok_sca
S
1
Segmento 35
ok_sca
SCARICO
S
1
Segmento 36
ok_sca
Com_medio
scarica
S
1
Segmento 37
sw_scarico
P
fine_scarico
Segmento 38
fine_scarico
scarica
R
1
ok_sca
R
1
SCARICO
R
1
perm_scarico
R
Segmento 40
fine_scarico
SUB_I
EN
ENO
num_medi
IN1
OUT
num_medi
+1
IN2
Segmento 41
num_medi
==I
+0
Com_medio
R
1

➢ C.6 Misur

Segmento 42
inizio misura
ok
perm_mis
ok_m
S
1
Segmento 43
ok_m
MISURAZIONE
S
1
Segmento 44
ok_m
s1_mis
s2_mis
s3_mis
T97
IN
TON
+500
PT
Segmento 45
T97
Basso
Segmento 46
ok_m
s1_mis
s2_mis
s3_mis
T98
IN
TON
+500
PT
Segmento 47
T98
Medio
Segmento 48
ok_m
s1_mis
s2_mis
s3_mis
T99
IN
TON
+500
PT
Segmento 49
T99
Alto

Segmento 50
Alto
fine_misura
Basso
Medio
Segmento 51
Basso
Com_basso
S
1
Segmento 52
Alto
Com_alto
S
1
Segmento 53
fine_misura
MISURAZIONE
R
1
ok_m
R
1
perm_mis
R
1
Segmento 54
Medio
ADD_I
EN
ENO
+1
IN1
OUT
num_medi
num_medi
IN2
Segmento 55
num_medi
>=I
+1
Com_medio
S
1

➢ C.7 Danger

Segmento 56 DANGER

emergenza

emergency

(S)

1

PERICOLO

(S)

1

Segmento 57

emergency

ruota_tavola

(R)

1

seleziona

(R)

1

scarica

(R)

1

➢ C.8 Errori

Segmento 62
ok_sel
emergency
T40
IN
TON
+60
PT
Segmento 63
T40
err_sel
S
1
err
S
1
Segmento 64
ok_sca
emergency
T41
IN
TON
+60
PT
Segmento 65
T41
err_sca
S
1
err
S
1
Segmento 66
err
ruota_tavola
R
1
seleziona
R
1
scarica
R
1
errore
S
1
Segmento 67
err
start
err_rot
R
1
err_mis
R
1
err_sel
R
1
err_sca
R
1

INDICE

➢ BIBLIOGRAFIA

- “PLC e automazione industriale” di Pasquale Chiacchio
- S7-200 Docoments Siemens

- www.itlug.org
- www.lego.com
- http://www.cs.brown.edu
- www.siemens.com
- www.Im-software.com/mlcad

➢ Software e Strumenti utilizzati

- First ISaGRAF 3.31F
- STEP 7 – Micro/WIN 32 V 3.2.3.17
- MLCAD Version 3.00

- PLC della serie Siemens S7 – 200

RINGRAZIAMENTI

Un doveroso e sentito ringraziamento va al prof. Alberto Leva e a Fabrizio Bragantini.

Un grazie particolare anche ai compagni d'avventura